On Supporting Group Decision Making by a Hybrid Method of the Method of Pairwise Comparisons and the Delphi Method

by

Grant G. O. Duncan

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	On Supporting Group Decision Making by a Hybrid Method of the Method of Pairwise Comparisons and the Delphi Method
Name of Candidate Nom du candidat	Duncan, Grant
Degree Diplôme	Master of Science

Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance	March 6, 2015

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Waldemar Koczkodaj
(Supervisor/Co-directeur de thèse)

Dr. Ryszard Janicki
(Co-supervisor/Co-directeur de thèse)

Dr. Francisco Diaz-Mitoma
(Committee member/Membre du comité)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. David Lesbarrères
M. David Lesbarrères
Acting Dean, Faculty of Graduate Studies
Doyen intérimaire, Faculté des études supérieures

Dr. Kevin Kam Fung Yuen
(External Examiner/Examinateur externe)

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

Ranking and prioritization are important endeavors that take place in a multitude of industrial, commercial and professional domains. The Delphi method is a systematic procedure for aggregating experts' opinions, particularly those that are predicated on qualitative or otherwise judgmental information. The method of pairwise comparisons provides an elegant means by which the qualities of criteria or alternatives can be compared, ranked, quantized and provides methods for the identification and resolution of inconsistencies. This thesis examines and proposes a means by which the two methods can be combined, in order to provide an easy to use system (which is then implemented as a reference implementation) for ranking and prioritization of a number of alternatives that is broadly applicable to any number of ranking and prioritization problems.

Keywords

Pairwise comparisons, The Delphi method, ranking, prioritization, judgmental information, inconsistency, ASP.NET.

Acknowledgments

I would like to thank Prof. W. W. Koczkodaj for the formulation of research this topic and his tireless assistance especially with the system design.

I would also like to thank to Dr. R. Hendel, Dr. R. Janicki and Dr. F. Diaz-Mitoma for their support and encouragement, my wife Jana Duncan for her love, support and invaluable aid, without which this thesis would never have been finished. Thanks as well to my employer, Health Sciences North, for their encouragement and allowing me the time to pursue this degree in between my work duties. I would like to also acknowledge and provide sincere thanks to my advisory and review committee for contributing their time and effort. Finally, special thanks to the Sudbury & District Brain Injury Association's board of directors for their invaluable help in testing and fine-tuning the PC Delphi implementation.

I would like to dedicate this thesis to my son, Grant Jay Duncan, who was born during it's writing.

Contents

List of Figures

List of Algorithms

List of Appendices

Preface

This thesis leverages the wisdom of Llull, the 13th century mystic and philosopher (Pairwise Comparisons), the 20th century invention of the RAND Corporation (Research ANd Development; a nonprofit global policy think tank), (the Delphi Method) and 21st century technology, the Cloud and the Web 2.0. The ranking and deriving of weights of a number of alternatives or criteria is a multifaceted problem that spans many domains. Numerous schemes exist which allow for one or more individuals to rate and rank alternatives according to some metric, typically by means of some sort of quantification based on some metric. In order to provide a collaborative approach to the problem, the synthesis of two methods will be explored: the method of Pairwise Comparisons and the Delphi method, called the PC Delphi. As well, a method of providing for the quantitative assessment of alternatives when compared on to another in pairs will be introduced. This method will help mitigate the frequently very difficult scaling issues that plague pairwise comparisons implementations; allowing the user to judge the comparison with their "gut feeling" rather than relying on an overt discrete comparison scale (which could result in frequently meaningless comparative values). The World Wide Web is used as the communications platform upon which the PC Delphi is based. Finally, a reference implementation will be provided, written in ASP.NET, C#, JavaScript, HTML5 and Transact-SQL, targeted at the .NET framework version 4.5. The reference implementation is given in it's entirety in Appendix 1; it represents a platform for further research and provides a template upon which similar solutions may be built (or the existing implementation extended).

1 The Method of Pairwise Comparisons

A fundamental process for deriving measurements is the direct comparisons of objects with regard to a property. Comparing objects in pairs for barter might have been used for as long as humans have existed. Comparing entities, abstract or physical, in pairs, is needed for any kind of measurement. The measurement process simplifies when a unit of measure (for example, a gram or a meter) exists. For abstract entities, such as environmental pollution or public safety, there is no unit. In such cases, pairwise comparisons (PC) are of considerable importance [1]. According to Herman & Koczkodaj [2], "[t]he pairwise comparisons methodology introduced by Thurston in 1927 [in his Law of Comparative Judgments] can be used as a powerful inference tool and knowledge acquisition technique in knowledge-based systems." Furthermore, "[t]he practical and theoretical virtue of the pairwise comparison methodology is it's simplicity" [2]. Beyond Thurston, the use of PC can be traced back to Fechner (1860) as a psychometric tool, to Cordocet (1785) and to Ramon Llull (around 1275), who is credited with discovering the Borda count and Cordocet criterion centuries before their namesakes [3, 4]. PC based on a ratio scale is fundamental to Saaty's *Analytic Hierarchy Process* (1980). PC as a method of decision resolution, especially in multiattribute decision making, is a well-known and intuitive technique that has since found applications in any number of problem domains, from tender evaluations and public procurement processes to hazardous waste disposal site selection and investment banking [2, 5].

1.1 Definition

Beginning with an expert (or team of experts) and a number of stimuli, denoted s_i (which could be concepts, physical objects, etc.), the expert would compare each of the stimuli in pairs, one-to-the other, producing a set of "coefficients $a_{ij} > 0$, which are meant to be a substitute for the quotients $\frac{s_i}{s_j}$ of the unknown (or even undefined) values of stimuli $s_i, s_j > 0$"; "[t]he quotients $\frac{s_i}{s_j}$ are also sometimes called the relative weights in the literature" [2]. A pairwise comparison matrix then, is defined as a square matrix $A \in \mathbb{R}^{\mathrm{nxn}}$, of values,$a_{i,j} > 0$, representing the expert's relative preference

of stimuli s_i over s_j (the i-th entity compared to the j-th entity):

$$A = \begin{pmatrix} 1 & a_{12} & \dots & a_{1n} \\ \frac{1}{a_{12}} & 1 & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{a_{1n}} & \frac{1}{a_{2n}} & \dots & 1 \end{pmatrix} \tag{1}$$

A is said to be reciprocal if $a_{i,j} = \frac{1}{aji}$ for every $i, j = 1, ..., n$. Note that the diagonal is 1, since $\frac{a_{ij}}{a_{ij}} = 1 \, \forall i, j = 1, ..., n$ [6].

1.2 Consistency and Inconsistency

As per Koczkodaj et. al.: "[o]ne of the important aspects of collaboration is inconsistency arising from different points of view on the same issue" [3]. Inconsistency indicates that "inaccuracy of some sort is present in the system" and is thus "related to the degree of contradictions existing in the assessments" [3]. "Inconsistency is often taken for a synonym of inaccuracy but it is a 'higher level' concept" and indicates "both the degree of inaccuracy and contradiction" [3] present in a PC system. Inconsistency is of central concern to the method of pairwise comparisons, and as such it's analysis, management and quantification is a priority task.

The simplest case of inconsistency occurs when given 3 stimuli, *A, B, C*; assuming that $A > B$,and $B > C$. Claiming then that $C > A$ is obviously inconsistent with our earlier assumptions, in which $A > C$ by transitivity of the > relational operator. A PC matrix A is said to be *consistent* (or transitive) if:

$$a_{ij} a_{jk} = a_{ik} \forall i, j, k = 1, ..., n \tag{2}$$

or put another way, A is consistent "if and only if there exists a positive n-vector w such that $a_{ij} = \frac{w_i}{w_j}, i, j = 1, ..., n$" [3]. It should also be noted that for a consistent matrix A, "the values w_i serve as priorities or implicit weights of the importance of alternatives" [3].

There are several means by which inconsistency (or consistency) within a PC system can be derived. One approach, proposed by Saaty is the *consistency index* (CI) (defined in the context of the analytic hierarchy process (AHP)):

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{3}$$

where $\lambda_{max}$is the largest eigenvalue of a $n \times n$ reciprocal pairwise comparison matrix. If a decision maker is perfectly consistent, then $\lambda_{max} = n$ therefore $CI = 0$, otherwise $\lambda_{max} > n$. To then measure the consistency, Saaty proposes the consistency ratio defined as:

$$CR = \frac{CI}{RI} \tag{4}$$

where RI is the average value of CI obtained from 500 positive reciprocal PC matrices "whose entries were randomly generated using the 1 to 9 scale" [7]. A value of CR under 0.10 is considered to indicate that the decision maker is sufficiently consistent.

Numerous criticisms of Saaty's methods exist within the literature. For instance, from [7]: "[w]hy ten percent?" and "[s]hould the cut-off rule be a function of the matrix size?". Koczkodaj in [8] raises similar concerns about the seemingly arbitrary 10% threshold, and also raises the concern of localization of the inconsistency within the PC system. Since an "eigenvalue is a global characteristic of a matrix (...) [by] examining it we cannot say which matrix element contributed to the increase in consistency" [8]. To say nothing of the computational complexity of calculating eigenvalues (for 500 positive reciprocal PC matrices, no less).

In contrast to Saaty's method, Koczkodaj has instead proposed an *inconsistency indicator* (II) for measuring and localizing inconsistency within a PC system in 1993. In [8], Koczkodaj defines consistency as "a measure of deviation from the nearest consistent basic reciprocal matrix". Given a vector of 3 components$[a,b,c]$, from (2) we can derive that $b = ac$ is true for each consistent reciprocal PC matrix. As per [8], "[w]e can always produce three consistent basic reciprocal matrices (represented by three vectors) by computing one coordinate from the combination of

the remaining two coordinates", the resultant three vectors being $\left[\frac{b}{c}, b, c\right]$,$[a, ac, c]$and $\left[a, b, \frac{b}{a}\right]$. Koczkodaj is then able to define his *consistency measure* (in the case of $n = 3$ Euclidean matrices) as follows [8]:

$$CM(a,b,c) = min\left(\frac{1}{a}\left|a - \frac{b}{c}\right|, \frac{1}{b}\left|b - ac\right|, \frac{1}{c}\left|c - \frac{b}{a}\right|\right) \tag{5}$$

Some of the stated advantages of this consistency measure include being easy to interpret, forming a better basis for selecting a threshold based on common sense and enabling the localization of the inconsistency.

Further work by Duszak and Koczkodaj in 1994 [9] extended the definition (5) for general *n*-ranked reciprocal matrices to the following:

$$CM(A) = max\{CM(a,b,c)\} \tag{6}$$

Finally, Bozoki and Rapcsak in [5] demonstrated that the above is equivalent to:

$$II = min\left\{\left|1 - \frac{b}{ac}\right|, \left|1 - \frac{ac}{b}\right|\right\} \tag{7}$$

Replacing *a, b, c* with the elements of the PC matrix and further simplifying as in [1], *A* (1) gives us the form of the equation that will be used throughout the remainder of this document:

$$II = 1 - min\left(\frac{a_{ij}}{a_{ik}a_{kj}}, \frac{a_{ik}a_{kj}}{a_{ij}}\right) \tag{8}$$

The three elements a_{ij}, a_{ik}, a_{kj}, are called a "triad". The overall inconsistency of a PC Matrix *A* is the maximum *II* over all the triads in *A*. As per Koczkodaj et. al. from [10], "[a]n acceptable threshold of inconsistency is 0.33 because it means that one judgment is not more than two grades of the scale 'different' from the remaining two judgments".

1.3 The Scaling Problem

Numerous authors have attempted to tackle the issue of how to provide an adequate scale by which pairwise comparison judgments may be evaluated. Luce and Edwards in [11] provide an analysis of Thurston's work, but as per Koczkodaj et. al. "[t]he bottom line is that subject quantitative assessments are not easy to provide. Not only is the dependance between the stimuli and their assessments usually nonlinear, but the exact nature of the non-linearity is unclear" [3]. Crawford and Williams in [12] characterize the issue by stating "[m]any methods have been developed for constructing scales of measurement based on subjective data. Several books and hundreds of articles have been written about these methods". Dehaene provides an analysis of the neurological basis of the Weber-Fechner law [13], as well as an analysis of Neider and Miller's work showing evidence for the logarithmic coding of numbers within the brain. Weber had discovered that "over a large dynamic range, and for many parameters, the threshold of discrimination between two stimuli increases linearly with stimulus intensity" [13], while Fechner postulated that "the external stimulus is scaled into a logarithmic internal representation of sensation" [13]. Other authors, such as Stevens and Shepard have theorized that the internal scale is a power function, or a multidimensional internal scaling (when applied to estimation), respectively.

According to [1], eigenvalue-based inconsistency tolerates errors of arbitrary value, hence they can no longer be considered correct. Koczkodaj introduced a five point scale in the 1990s. At the time, it was regarded to "better fits the heuristic of 'off by one grade or less' for the acceptable level of the distance-based inconsistency". However, a mathematical theory in [3] based on the Fulop constant, provides strong reasons to use an even smaller scale, from 1 to 3. The rational for a smaller scale being that it is "expected to generate a smaller error, for example by mitigating the deviation from non-linearity" [3]. The five point scale is also used by the Likert scale, a psychometric scale often used in questionnaires (familiar to anybody who has filled in a "customer satisfaction" style survey at some point in their lives). By contrast, Saaty, in his Analysis Hierarchy Process advocates for the use of an odd ratio scale [14] and a "fundamental" scale ranking intensity or importance by using the odd integers 1,3,5,7,9 and even 2,4,6,8 as values "between", with

fractions allowed only between 1 and 2.

Clearly, assigning a quantitative value to a subject assessment is fraught with difficulties, to say nothing of the apparent arbitrary nature of such assignments (and the values they represent). For example, if asked to rate how much more somebody might enjoy watching television over shoveling the walkway, what would be the meaning of stating "twice as much" versus "three times as much"? Neither provides a useful metric by which to gauge which activity is more enjoyable by how much; the numbers in this sense are meaningless.

1.4 Comparisons by Sliders

There is thus a clear and universal problem of attempting to establish a scale of measurement for comparisons of subjective data. It is not simply the scale that can confound an estimator, but also the problem of meaning. What does it mean to say that one prefers alternative 1 to alternative 2 by a factor of 5? How about if one were to use a ratio scale (as is preferred by some authors, notably Saaty) and say that the preference is now 5:1? There is an inherent difficulty in assigning meaning to quantitative judgments of subjective attributes. And yet, one can easily imagine any number of qualitative methods by which subjectivity can be gauged. Attempting to determine which of two small objects is heavier may invariably lead to taking each object in one hand, and balancing between them in order to determine their relative weights. Experiments by Koczkodaj and others have shown that individuals are able to accurately gauge the differences in length between two lines by comparing them in pairs; in the context of computer aided decision making systems, these experiments suggest a readily available and easy to implement user interface element - the slider.

The use of a slider to gauge the comparison of subjective qualities between two alternatives (when compared in pairs) allows for a clearly defined visual representation. The handle of the slider can be moved toward the alternative in proportion to the estimated ratio of the weight of the compared attribute. An additional visual aid is created via the use of a word-cloud like technique; the labels of the alternatives are scaled according to the log of the comparisons, inspired by the Weber-Fechner law's concept of a just noticeable difference - the just noticeable difference between

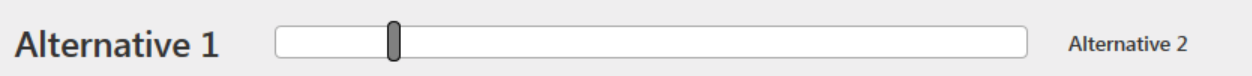

Figure 1: A slider and it's associated labels

two stimuli is proportional to the magnitude of the two stimuli. Multiple discrimination tasks such as detecting changes in the length of a line on a screen "all obey Weber's law in that the observed values need to change by at least some small but constant proportion of the current value (...)" [15]. Figure 1 displays a typical slider and it's associated labels. Notice how it is clear that Alternative 1 is preferred to Alternative 2, however there are is no direct quantification of the comparison.

2 The Delphi Method

The Delphi method "originated in a series of studies that the RAND corporation conducted in the 1950s" whose "objective was to develop a technique to obtain the most reliable consensus of a group of experts" [16, 17]. Listone and Turoff [18] characterized it as a "method for structuring a group communication process" to enable a "group of individuals, as a whole, to deal with a complex problem". In order to accomplish this, they determined four defining attributes:

- Some feedback of individuals contributions,
- Some assessment of the group judgment of view,
- Some opportunity to individually revise views, and
- Some degree of anonymity for the individual responses.

The Delphi method is typically used in cases where "judgmental information is indispensable" [19], often times via the use of questionnaires in order to "avoid direct confrontation of the experts". The Delphi method is used extensively in Forecasting [20] (hence the name "Delphi", inspired by the Oracle of Delphi). A "Delphi" (a colloquial appellation of the implementation of a Delphi method session) is predicated on the hypothesis that through an iterative process the experts will eventually converge on a consensus opinion.

The Delphi method has been applied to a "wide variety of situations as a tool for expert problem solving". Aside from forecasting, the Delphi method is also used in concept and framework development, typically involving a two step process beginning with the identification of a set of concepts and ending with classification or taxonomy development [16]. Okoli and Pawlowski give an overview of studies conducted in both application domains [16].

The true strength of the Delphi method is it's ability to represent the so-called *wisdom of the crowd*; from [16]: "[s]tudies have consistently shown that for questions requiring expert judgment, the average of individual responses is inferior to the averages produced by group decision processes; research has explicitly shown that the Delphi method bears this out". In terms of the data

retrieved by the application of the Delphi method, "Delphi studies inherently provide richer data because of their multiple iterations and their response revisions due to feedback" [16]. Note that the Delphi method is not always used as a consensus tool, for instance the *Policy Delphi* emphasizes the identification of differing opinions and divergent responses mediated through a Delphi-like process [21].

2.0.1 Implementing a Delphi

Before beginning a Delphi, it is important to ensure the appropriateness of the technique to the problem at hand. Linstone [18] identifies two appropriate circumstances: "the problem does not lend itself to precise analytical techniques but can benefit from subjective judgment on a collective basis" and "individuals who need to interact cannot be brought together in a face-to-face exchange because of time or cost constraints". Once the problem has been identified, a typical Delphi method implementation (also called a *nominal* Delphi [22]) has a number of steps that can be broadly enumerated as follows:

1. Identification of the group members, and the facilitator (also sometimes called the moderator or administrator).

2. First questionnaire: each group member identifies the issues, criteria or alternatives under consideration and submits this information to the facilitator, who then pares down the list into a manageable summary.

3. Second questionnaire: group members rate or rank the list items (typically submitted back to the group members in randomized order).

4. Subsequent questionnaires: group members are presented with the results of the second questionnaire and the group's consensus metrics. Dissenting members have a chance to justify their dissensions, and all participants have the opportunity to revise their assessments. This continues iteratively until some stopping criteria is reached, either via a pre-destined number of steps or after a number of consensus criteria or metrics are achieved.

5. The final results are submitted to the group for acceptance and ratification.

Typically three or four rounds are suggested in order maximize the participant group's goodwill and willingness to participate, though there is no hard and fast rule as to precisely how many rounds should be performed [22, 21]. According to Worthen and Sanders, the iterative step can "continue for several more rounds, but the payoff usually begins to diminish quickly after the third round" [23]. Note as well the importance of ensuring that the group members are a "panel of subject-matter experts (SME)" [24], since fundamentally the outcome of a Delphi is "nothing but opinion; the results of the [process] are only as valid as the opinions of the experts who make up the panel" [22]. While panel selection should be somewhat randomized in order to ensure a broad scope of opinion and minimize biases, "[p]anel selection might not be random because, in some research fields, there might be very few SMEs; thus, one might select all known SMEs" [24]. In terms of the size of the participant group, Loo recommends "15-30 carefully selected SMEs (...) for a heterogeneous population and as few as five or ten for homogeneous populations" [24].

2.0.2 Strengths and Limitations of the Delphi Method

Yousuf identified the following strengths of the Delphi method [22]:

1. "The problem does not lend itself to precise analytical techniques but can benefit from subjective judgments on a collective basis."

2. "The individuals needed to contribute to the examination of a broad or complex problem have no history of adequate communication and may represent diverse backgrounds with respect to experience and expertise."

3. "More individuals are needed than can effectively interact in a face-to-face exchange."

4. "Time and cost make frequent group meetings infeasible."

5. "The efficiency of face-to-face meetings can be increases by a supplemental group communication process."

6. “Disagreements among individuals are so severe or politically unpalatable that the communication process must be refereed and/or anonymity assured.”

7. “The heterogeneity of the participants must be preserved to assure validity of the results.”

Clearly, the method provides a structured and regimented technique with which to elicit consensus opinion from a group of experts. Yousuf references a number of authors who have researched the Delphi when compared against other group or panel decision making techniques and in his words have found that it has “application in reliability and many advantages”. Further strengths identified by Yousuf include the fact that the “technique is simple to use”, “[a]dvanced mathematical skills are not necessary for design, implementation and analysis of a Delphi project” and due to the inherent anonymity or confidentiality of the technique, it can overcome “many barriers to communication” [22]. It should also be emphasized that a perceived strength of the Delphi method is that it can help prevent “groupthink” [22]. Loo in [24] makes mention of the Delphi technique offering several advantages over other group decision making methods such as the nominal group technique and interacting group method by noting that because “idea generation in the Delphi is individual based, anonymous and independent (...) panel members are not swayed by group pressures or vocal members as easily”. Note as well that “the use of successive rounds in a Delphi enables the moderator to build upon earlier results and to maintain focus in the study” [24].

However, the Delphi method is certainly not without criticisms. Loo states that “as early as 1975, Sackman raised a series of questions about the scientific basis of the Delphi method” [24], though in his opinion “a large body of literature has accumulated to demonstrate the usefulness of the method when well-designed and executed” [24]. Some of the major criticisms center around the following points [24]:

1. The sample size is seen as too small. In order to minimize error and bias and to maximize statistical reliability, a much larger sample size is typically required.

2. Reliability of the measures derived from the judgments is often seen as questionable and unreliable, given that different panels can give differing responses and in some instances it is

claimed that the consensus thus achieved may be due to pressure or willingness to conform rather than a genuine convergence of opinion.

Additionally, Yousuf notes that the Delphi method is often criticized due to it "(a) being unscientific; (b) having a low level reliability of judgments among experts and therefore dependency of forecasts on the particular judges selected; (c) the sensitivity of results to ambiguity in the questionnaire that is used for data collection in each round; and (d) the difficulty in assessing the degree of expertise incorporated into the forecast" [22]. However, he also notes that the "Delphi is a method of last resort in dealing with extremely complex problems for which there are no adequate models. Sometimes reliance on intuitive judgment is not just a temporary expedient but in fact a mandatory requirement" [22].

Logically speaking, no group decision making or consensus system is going to be completely reliable, deterministic, or even necessarily correct. Loo recommends that in order to diminish some of these perceived weaknesses researchers should "consider using a triangulation of methods (...) [involving] the use of two or more complementary methods of data collection with the expectation that the results will converge" and that "[f]or many situations, researchers may find the combination of a Delphi study and a survey with two independent samples useful and practical" [24].

2.1 The "Ranking-Type" Delphi

A variant of the Delphi method [20] that "has been used extensively in information systems research to identify and rank key issues for management actions" is the so-called "ranking-type" Delphi. Schmidt provides the most comprehensive overview of a typical implementation (via questionnaires administered by Couger in 1988) and subsequently suggests some necessary refinements to the process.

2.1.1 A Typical "Ranking-Type" Delphi

In the first round, respondents were asked to list a number alternatives, in rank order; participants were furthermore encouraged to include their rationale for inclusion for each alternative. In the

second round, participants were asked to submit a ranked, consolidated list of alternatives. Additionally Couger asked respondents to submit their ranking as a top 20; ties were disallowed. These submitted rankings were then consolidated into an ordered list of "top 10" and "bottom 10" alternatives. The third (and final) round had the participants ranking their top 10 choices, while all others were given a ranking of zero; ties were again disallowed.

After assembling the results of the third round, the mean ranks of the top 10 alternatives in the second and third rounds were compared to the overall mean ranks of all the alternatives (over all the rounds). Ties were broken via the comparison of the size of the alternatives' standard deviations. Furthermore, Couger measured consensus using Kendall's coefficient of concordance, *W*, which provides an indication as to whether consensus has been reached, whether consensus is increasing, and the strength of such consensus. Kendall's coefficient measures agreement via the application of a least squares solution. According to Schmidt it is the "most popular method for this purpose, mainly due to it's simplicity of application" [20]. Other metrics to measure consensus have been suggested (such as by Cook and Seiford [20]), but according to Schmidt these produce less-than-optimal solutions for application to a Delphi. It should be noted that while Couger utilized Kendall's coefficient in each round subsequent to the first, other researchers (such as Brancheu and Wetherbe) opted for the use of Kendall's coefficient in the final round of their surveys.

2.1.2 Schmidt's Refined Process

Schmidt suggests the following as an idealized and refined procedure for the implementation of a ranking-type Delphi: "(1) discovery of issues, (2) determining the most important issues, and (3) ranking the issues". In the first phase, all respondents are encouraged to submit at least 6 important issues or alternatives [20] along with an adequate description of these alternatives since "respondents are likely to raise the same issue using different terms"[20]. These would then be consolidated into a single list where each description is an aggregate of the submitted issues' descriptions. Schmidt is careful to indicate that at this stage "there is no basis of claim that a valid,

consolidated list has been produced" [20]. If the Delphi encompasses comparing the responses from multiple disparate groups, it is critical that that all groups "must participate collectively in the first phase. Otherwise the researcher will face great difficulty in subjectively mapping the group's independently ranked items for comparison" [20]. This phase should be repeated until major differences in respondents suggested issues are resolved.

The second phase consists of the researcher paring down the submitted lists so they can be "meaningfully ranked" [20]. While Schmidt makes reference to some researchers combining this phase with the third phase, he cautions that in cases of many issues or alternatives the sheer number may hinder the ranking phase. It is suggested that the researcher disseminate a randomly ordered listing of the consolidated issues list from the first phase to each of the participants wherein they are asked to "independently select at least 10% (or more if the list contains less than 100 items) of the [alternatives] as the most important" [20]. Issues that were not selected by a majority of the respondents are eliminated. It is further suggested that the list of issues be kept at around 20 items or less (and if the submitted issues list is already 20 or less than this phase is omitted and the third phase can begin), though in reality any arbitrary target for the size of the list can be given; the justification being that "setting an arbitrary size for the list forces the result" [20].

The third, and final phase of Schmidt's ranking-type Delphi asks the participants to rank all the remaining alternatives. It is suggested that whenever possible the respondents are asked to avoid ties, or the researcher should compensate for ties while calculating the strength of the consensus (via Kendall's W). This will continue in an iterative fashion, allowing the respondents to revise their rankings, until the stopping criteria are reached. The stopping criteria are highly dependent on the feasibility of continuing (participant fatigue, logistics and resources) and the potential gains (for instance, a higher consensus rating).

Feedback is of essential and critical importance to any Delphi. Graphical displays, tables and descriptive statistics can be used in order to attempt to represent the information such that it is comprehensible to it's users. Schmidt emphasizes the importance of reporting "round-by-round levels of consensus and any other measures of association pertinent to the research hypotheses"

[20].

2.2 Arrow's Impossibility Theorem

In 1950 Kenneth Arrow of Stanford University published his seminal paper "A Difficulty in the Concept of Social Welfare" [25]. In it, he proved that, given a decision making body with two or more members deciding among at least three alternatives, there is no rank voting method (referred to by Arrow as a "social welfare function", the decision making body is referred to as a "society") that can be devised that satisfies all of the following conditions:

1. Universality of the social welfare function (from Arrow: "[t]he social welfare function is defined for every admissible pair of individual orderings" [25]),

2. Positive association of social and individual values ("the social ordering responds positively to alterations in individual values or at least not negatively" [25]),

3. The independence of irrelevant alternatives (sometimes referred to as pairwise independence - "the choice made by society from any given set of alternatives should be independent of the very existence of alternatives outside the given set" [25]),

4. Non-imposition (which Arrow referred to as "the condition of citizens' sovereignty": "[t]he social welfare function is not to be imposed" [25]), and

5. Non-dictatorial ("social choices are not to be based solely on the preferences of one man" [25]).

A later version of the theorem replaced the positive association and non-imposition conditions with "Pareto efficiency"; such that unanimity of the candidates' rankings implies the same societal rankings.

The implications of Arrow's theorem to any system that produces a rank ordering of it's alternatives is sobering; in effect no rank ordering system can comply with the above requirements and deterministically (and accurately) reflect the "true" ranking as desired by the group. Furthermore,

Arrow demonstrated in "Social choice in individual values" that with any method used to produce a consensual choice, the choice is influenced by the method. It is important, then, to take special note that the solution to (in particular) a ranking-type Delphi is thus imposed by the method and may not necessarily reflect the most optimal solution. However, Schmidt claims that the "iterative approach [of a Delphi] lessens this effect by allowing panelists to revise their choices" [20].

2.3 The Web-based Delphi

In 1995, Hartman and Baldwin [26] introduced what they termed the "Modified Delphi Method", which is a "consultative process that uses computer technology combined with a survey to obtain constructive participation in the input and consensus process". As was previously noted, *nominal* Delphis may be criticized for the a number of reasons. From [26]:

1. The panel used, of necessity, is normally limited to a few participants.
2. Because of the limited number of participants, their biases may influence the outcome of the study.
3. Selection of the panel requires great care to ensure that a representative group has been used.
4. Interaction between participants is limited or non-existent depending on the procedure adopted by the researcher.
5. One panelist may dominate the direction of opinion and advice by virtue of their personality, position of influence or other factor.

Hartman and Baldwin believed they could "accelerate" and "improve" the process by doing two things: (1) they could extend the panel size to a larger group, as long as "the number of iterations could be reduced and the obtained opinions or data could be analyzed effectively", and, (2) the "commentary could be obtained in a manner that allowed panelists to comment on each other's comments at the same time as making original comments, thus obtaining some of the advantages of the iterative process of the traditional Delphi Method" [26]. In their study (in contracting in

the construction industry), their participants concurrently completed questionnaires on networked personal computers; all comments were shared by all participants, stored on network's file server and remained completely anonymous. According to their study, the participants observed the following [26]:

- The process is equivalent to a significant number of meetings.
- It is an effective way in which to get ideas on the table.
- It allows candid input and more valid conclusions.
- It creates a focused environment in which to address the issues at hand.
- The process encourages response. This is of a particular value to those who feel inhibited in giving opinions in a group environment.
- It is a good working example of how technology can speed up the interchange of ideas.
- The experience is fun.

The techniques they used had been developed by the Faculty of Management at the University of Calgary. According to Hartman and Baldwin it is "frequently used by business in brainstorming session, often on sensitive matters, in which anonymity and an equal voice for all participants are an asset to the consultative process" [26]. They are quick to point out that new process is "faster and requires fewer iterations of reviews with the panel of experts" [26], a clear advantage over the *nominal* Delphis.

Hartman and Baldwin are certainly not the only researchers who have implemented and/or experimented with web-based Delphis. Indeed, in the modern era, the use of a web-based application in order to implement a Delphi-like process is both obvious and expected. A simple search of either The Web of Science or Google shows an abundance of resources when using the search term "web based Delphi"; 240 papers and over 3.9 million results respectively, with the process defined under a variety of names such as the "Real-Time Delphi", "Electronic Delphi", "Computer Delphi", "Computer-aided Delphi", etc. The advantages to using a modern technological approach are apparent: asynchronicity and anonymity. According to Turoff and Hiltz, an asynchronous Delphi

has two key characteristics [27], “a person may choose to participate in the group communication process when they feel they want to” and “a person may choose to contribute to that aspect of the problem to which they feel best able to contribute”. Asynchronicity allows a participant to contribute at any time, unlike a traditional Delphi whereby the participants are “forced into lock-step” [27] with one-another. Allowing an individual freedom to contribute only to those Delphi steps where they feel appropriately equipped allows “individuals with differing perspectives and/or differing cognitive abilities to contribute to the parts of a complex problem for which they have both the appropriate knowledge and appropriate problem solving skills” [27]. While anonymity is a part of the *nominal* Delphi, computational techniques and role-based administrative and data management systems allow for multiple “variations in anonymity not possible in paper and pencil environments” [27]. As stated by Turoff and Hiltz, “[i]n essence, the objective of anonymity is to allow the introduction and evaluation of ideas and concepts by removing some of the common biases normally occurring in the face-to-face group process”, a goal which is easily achievable with even a simple web-based application.

However, it should still be noted that a Delphi process, regardless of implementation, is still fundamentally a collaborative endeavor, and as such would still require the use of a facilitator and some sort of structured process in order to be achievable [27, 22, 20]. Numerous online implementations of the *nominal* Delphi and it’s common variants exist online.

3 The PC Delphi

According to [12] "[t]he quantification of subjective data is essential for dealing with a wide class of problems whose solutions by other methods would be extremely difficult or impossible. Such problem are often amorphous and vaguely stated". These problems typically "lack any well-defined, scalar-values measures of merit" [12], and even given that a subjective facet could be quantified, "the collection of relevant objective data might be prohibitively expensive or impossible" [12]. Oftentimes the best available information is subjective, not quantitative, and thus the need for a quantitative, formalized methodology may seem counter-intuitive. As was covered in a previous chapter, the Delphi method is typically used in cases where "judgmental information is indispensable" [19]. Formalizing the acquisition and dispensation of such knowledge "gives structure and definition to an amorphous mass of data" [12], "permits sensitivity analysis on alternative judgments" [12] and is repeatable, thereby providing "an audit trail" of the considerations undertaken by the experts [12].

3.1 Synthesizing the Delphi Method with the Method of Pairwise Comparisons

3.1.1 Previous Work

The synthesis of the Delphi method with the method of Pairwise Comparisons described herein will be coined the "PC Delphi" or pairwise comparison Delphi. The usage of the method of pairwise comparisons (PC) in conjunction with the Delphi method seems like an obvious extension and application of both methods, but in reality is rarely seen in the literature. When implemented, the emphasis is on the use of PC in conjunction with Saaty's analytic hierarchy process (AHP), with the Delphi method being most often leveraged as the tool with which expert's opinions are elicited.

Xiajun Yang et. al. demonstrated their use of the AHP over the course of a one iteration Delphi in handling "randomness and fuzziness of individual judgments" with the Cloud model in estimating the relative area sizes of six provinces in China [28]. Their method consisted of

obtaining "the individual Cloud comparison matrix from the interval judgments and define the consistency index (CI) of the Cloud matrix", followed by a "one iteration Delphi method (...) applied to reduce individual mistakes", calculating the "individual Cloud weights (...) from the positive reciprocal Cloud matrices" in order to obtain the weights (via a geometric mean) and ending with a discussion of the ranking alternatives [28]. According to the authors their method can "reduce mistakes and uncertainties, and thus improve DMer's [sic] subjective judgments".

A similar model was utilized by Chung-Min Wu et. al. in [29], where they utilized what they called the *fuzzy* Delphi, the analytic network process (ANP) and TOPSIS in order to optimize the selection process criteria and evaluation of a supplier in a supply chain. They begin by using the fuzzy Delphi, a methodology in which subjective assessments are transformed via statistical analysis and fuzzy operations and continued until a reasonable level of convergence; a membership function and a min-max algorithm is used to triangulate expert opinions, to identify the selection criteria, while the geometric mean of each criteria is used to denote the consensus of the experts' evaluations. Following this, the weights of the selected criteria are determined via the ANP (in order to solve the problem of selection criteria interdependency). Finally, the TOPSIS method (a method proposed by Hwang and Yoon in 1981 which "enables decision makers to determine the positive ideal (A^*) and negative ideal (A^-) solution" [29]) is used to rank the alternatives. The authors assert that through this method "better decisions in supplier selected" can be achieved.

The fuzzy Delphi and the method of pairwise comparisons was also used by Vatalis et. al., in [30] and used to determine an environmental quality index (EQI) related to coal mining activities in Greece. A 12 member panel of "specialized environmental engineers, land use planners, ecologists, managers of local public utilities and regulatory officials" [30] were surveyed in order to determine the list and weights of environmental parameters. Their study was used to demonstrate an increasing EQI trend (by 2.11 per year) when computed over the period of 1983 to 1998. The authors make note that while their method was designed for coal mining and power plant facilities, the methodology could be applied to other cases as well when measuring environmental quality and it's trends.

A multi-criteria decision making model utilizing the Delphi method and the AHP was applied to classifying wood products with respect to their impacts on the environment in [31]. The parameters were organized into a multilevel hierarchical structure as per the AHP; 52 experts where then polled in a 3 round Delphi process whose assessments were used in the aggregation of the parameter values. The authors allowed two modes of assessment input: directly via numerical form or indirectly via pairwise comparisons of the alternatives. The model was specifically developed to address local conditions within the Republic of Slovenia; consequently all the experts were Slovenian and the authors caution that their model would require some adaptation if used in other jurisdictions.

Kocaoglu et. al., introduce the use of a modified Delphi method (which they call "snow ball") and pairwise comparison analysis in order to determine an indicator of the diffusion of new technology in [32]. They began by determining an initial list of experts using bibliometric analysis followed by a Delphi which was used to increase the number of experts until a desired number was identified. The group then used PC to define to rankings between the factors.

In [33] Medoza and Prabhu utilize multiple criteria analysis to assess the criteria and indicators adapted for a given forest management unit. The Delphi method and pairwise comparisons are used to assess the importance and prioritization of their given indicators. According to the authors, the Delphi method was selected "because it seems to be an attractive alternative model because of it's ability to accommodate multiple stakeholders and include their opinions in a democratic decision-making context" [33]. Their desire to use PC was prompted by it's "ability to measure the degree of inconsistency relative to the one-on-one comparisons" [33]. Two groups were asked to compare the indicators in pairs, one using the Delphi method and the other using a vote-by-consensus method. They found that the group using the Delphi method produced results that are more inaccurate (and thus less reliable), as opposed to the group using the vote-by-consensus method. However, it should be indicated that according to the authors "[d]ue to time constraints, it was not possible for group I to iterate the process to take advantage of the full capabilities of the pairwise comparison method using AHP" [33], which unfortunately undermines the usefulness of

the combination of the two approaches.

As was seen, the previous work performed by a variety of authors tended to achieve some success. By examining the (albeit sparse) previous work on the topic, the following trends emerge:

1. The Delphi method is typically applied in order to elicit expert opinions, or the fuzzy Delphi is used to help refine experts' numerical judgments over the course of 3 or fewer rounds (typically a single round).

2. The AHP or the ANP are used in order to evaluate the pairwise comparison systems, in a variety of situations.

Existing methods, frequently characterized by an ad-hoc formalism and approach to the usage of the methodologies, can be adapted in order to produce what will be hereafter termed the *PC Delphi*.

3.1.2 An Alternative Approach: The PC Delphi Process

A commonly referenced strength of the method of pairwise comparisons is its capability for representing and processing subjectivity by allowing "one to represent such subjective assessments and to process them by analyzing, quantifying and identifying the inconsistencies" [3]. Furthermore, "[t]he consistency-driven approach incorporates the reasonable assumption that by finding the most inconsistent assessments, one is able to reconsider his/her opinions" [34]. Managing inconsistency is "[o]ne of the major challenges for collective intelligence (...) which is unavoidable whenever subjective assessments are involved" [3]. The Delphi method can provide the mechanism by which an expert is able to re-evaluate their assessments, in order to minimize the inconsistency in their judgments, while also providing the benefits inherent therein (see "The Delphi Method"). Finally, in order to disseminate and coordinate the information, a well established communications platform that offers a degree of anonymity is required; the choice is obvious: the World Wide Web, otherwise commonly referred to as the internet. Thus, by using the method of pairwise comparisons to rate alternatives or criteria, in conjunction with the Delphi method's re-assessment mechanisms, a powerful, online tool can be built that would enable experts to collaborate on the

assessment of such alternatives, while also allowing for the re-assessment of those judgments in subsequent rounds.

Integrating the Delphi Method

Recall from chapter 3 that there are four characteristics that define a Delphi:

- Some feedback of individuals contributions,
- Some assessment of the group judgment of view,
- Some opportunity to individually revise views, and
- Some degree of anonymity for the individual responses.

Since the PC Delphi is primarily a ranking-type Delphi, feedback is inferred by the group rankings versus the individual rankings (as aggregated by the system) and group inconsistency indicator readings (note that when inconsistency indicators are discussed in the context of . Koczkodaj's inconsistency indicator allows for the localization of the inconsistent triads, which provides an easy to use and reconcile feedback mechanism.

Assessments of the group judgment are achieved using both the current aggregated rankings and weighting, as well as the overall group inconsistency. The ability to revise views and rankings occurs on a round-by-round basis. If a participant wishes to revise their views, they have the ability to on each subsequent round. A method to indicate previous responses (both group and individual) is suggested and can be accomplished by showing the previous selections on the slider itself. Anonymity is handled by any standard anonymity methods available to web developers. Unless specifically enabled, the web is an essentially anonymous communications system - barring any tracking that may be enabled by default on a given hosting platform (such as IP and machine tracking).

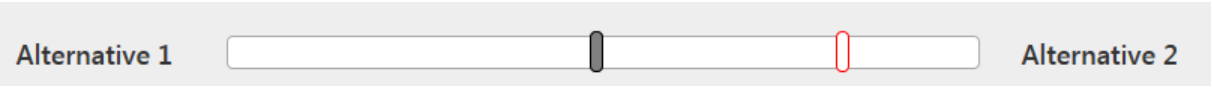

Figure 2: A Slider showing the previous value

Measuring Consensus with Koczkodaj's Inconsistency Indicator

Consensus decision making must be differentiated from decision making by unanimity - whereas unanimous decision making requires that all participants agree to the outcome, consensus decision making requires only that the participants give consent to the outcome; dissent therefore plays an important role. A consenting participant may not believe that the outcome is their best choice, but ultimately has given their consent to the outcome. Consensus decision making then is a decision making model of degrees rather than absolutes. Many methods exist for quantifying the degree of consensus and for resolving disputes that may arise - the most obvious of these are commonly known (and utilized) voting techniques: majority voting, preference voting and their ilk. A very well known technique (found in a number of textbooks) is Kendall's coefficient of concordance, or Kendall's W, which provides a measure of the agreement among several judges assessing a set of objects (criteria, alternatives,...). Kendall's W has found broad use in a number of diverse fields, for instance Legendre proposes it's use "to identify groups of significantly associated species in field survey data" [35] - a classical community ecology problem, while as seen in the Delphi chapter, Schmidt et. al. use it in it's classical formulation as a pure measure of concordance in group decision making. Other methods also exist, for instance in [36] Holey et.al. use simple statistics to measure consensus as a means of determining the stopping criteria in a Delphi, in particular by examining the convergence properties of the application of descriptive statistics and Kappa calculations over the course of a number of Delphi rounds.

This paper proposes that Koczkodaj's Inconsistency Indicator be used as a measure of consensus, when in the context of a PC Delphi. Recall that inconsistency is a measure of the distance from the fully consistent PC matrix, given three or more alternatives. Koczkodaj's Inconsistency Indicator provides for not only the quantification of that inconsistency, but also the localization to a particular triad. Given that an iterative pairwise comparison process should strive for reducing inconsistency (measured to some heuristic), then logically measuring this inconsistency in an aggregated PC matrix would provide a metric by which one could gauge consensus - the less inconsistency that exists, the more consistent are the aggregated ratings of the alternatives. Thus,

given a known heuristic (in the case of the PC Delphi this heuristic is set to around 1/3), it will be stated that sufficient consensus is achieved when the aggregated inconsistency index is less than or equal to it. The use of this metric also provides for some degree of dissent: triads showing inconsistency indicator values greater than zero (full consistency) clearly demonstrate that there are conflicts with the inter-rankings between these alternatives. These conflicts provide a measure of dissent and may indicate triads of comparisons requiring further study.

Handling Outliers

Similarly to how consensus is gauged, outliers are identified with Koczkodaj's Inconsistency Indicator. Recall that the global inconsistency of a PC system is equal to the inconsistency of the maximally inconsistent triad. During a PC Delphi, those submissions having a global inconsistency greater than a certain threshold, are able to be removed (or at least identified), as these could be considered outliers. It is suggested that the Delphi coordinator be given the ability to review the results with and without outliers. Should the rankings change drastically, then it is suggested that these outliers be removed from the final rankings calculations.

The PC Delphi Defined

The PC Delphi will now be defined. A reference implementation (in ASP.NET, C#, TSQL and JavaScript) is provided in the appendices.

Step 1: Select a facilitator

The facilitator will direct the Delphi, and manage any feedback or coordination tasks. The choice of facilitator should be an individual who is knowledgeable about the PC Delphi specifically, but also who has knowledge about the Delphi in general.

Step 2: Select a panel of experts

As was mentioned in chapter 3, when choosing a panel of experts note the importance of ensuring that the group members are a “panel of subject-matter experts (SME)” [24], since fundamentally the outcome of a Delphi is “nothing but opinion; the results of the [process] are only as valid as the opinions of the experts who make up the panel” [22].

Step 3: Selecting the alternatives or the criteria

Existing methods for criteria selection have been well-developed, and include polling the panel of experts, or pre-establishing them based on the nature of the problem. Given a large number of criteria, a "pre-ranking" process can take place, using the PC Delphi to determine a first-pass ranking using Koczkodaj’s simplified pairwise comparison technique, dropping those items that fall below a certain ranking. It should be noted that the PC Delphi is primarily a ranking Delphi, and as such this document will not provide additional information on how to establish the criteria for ranking.

Step 4: Perform the first round of ratings

The first round of ratings commences with all willing participants accessing the system and comparing the alternatives, one to each other, in pairs by sliding the handle of the slider toward the alternative that has more of the desired property or attribute being judged. The amount to slide should be gauged by the participant based on their “gut” feeling, using both the slider handle’s position and the relative sizes of the alternative’s labels as visual aids. Once the user is satisfied with their choice, they submit the results to the system.

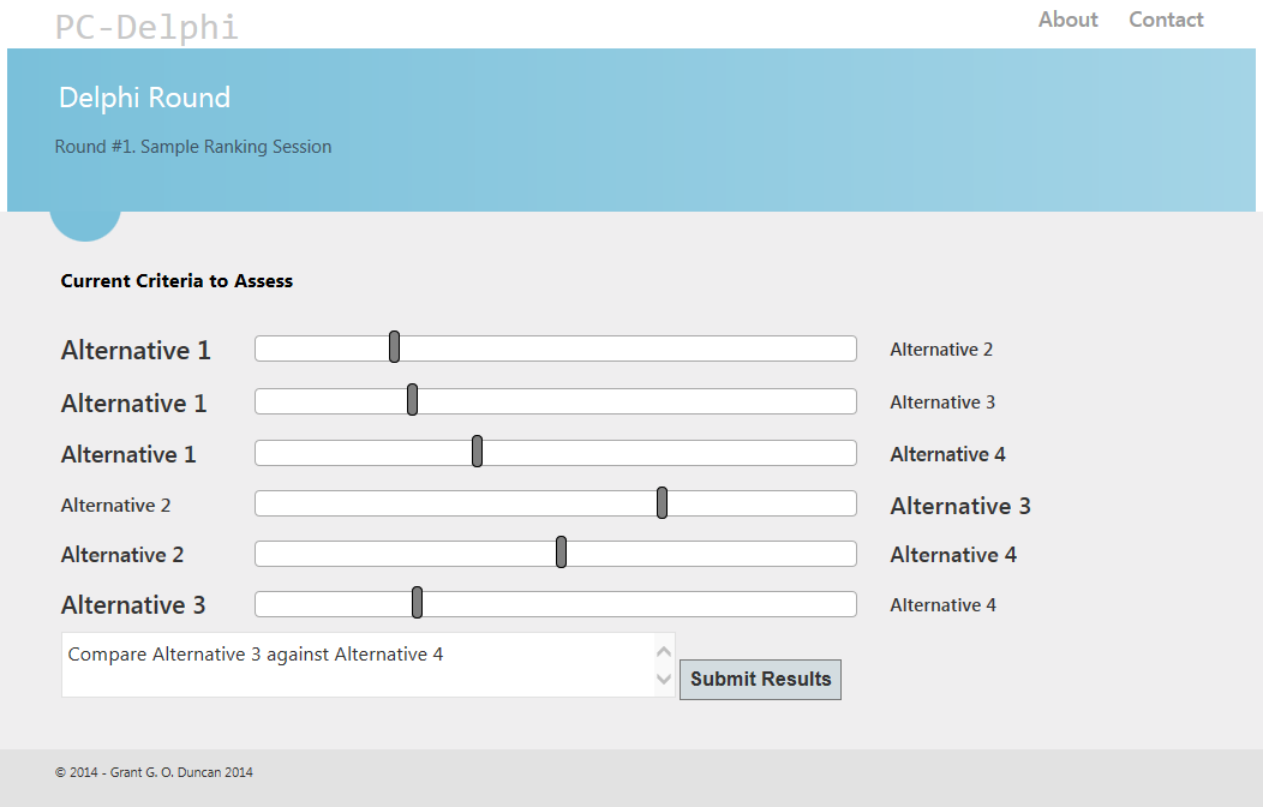

Figure 3: First Round Ranking Screen

At this phase of the process, only the administrative user(s) - i.e.: the Delphi facilitator and perhaps one or more designates - are shown the overall round 1 results. All other participants are redirected to a page thanking them for their participation in round 1. It should also be noted that a concrete time limit be provided to all the participants. The time limit would be dependent on the nature of the problem domain.

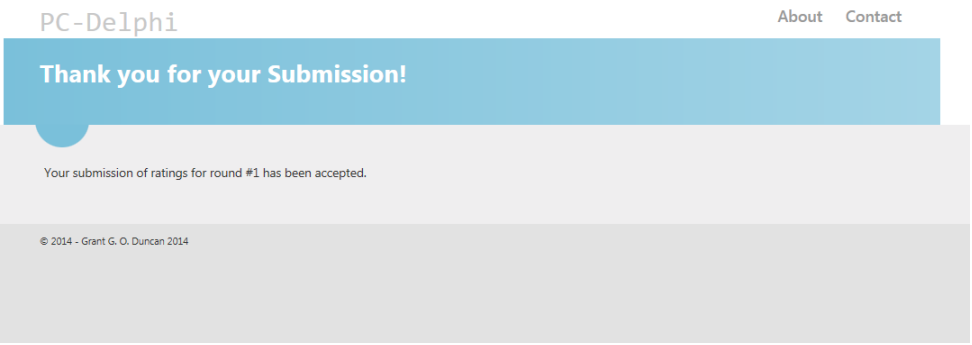

Figure 4: First Round End Screen for non-admin users

At this phase the first tests for consensus are performed - Koczkodaj's inconsistency indicator gives an indication of the measure of consensus as detailed previously, while also highlighting the triad that is causing the most dispute. Even if perfect consensus is reached however another round must be performed.

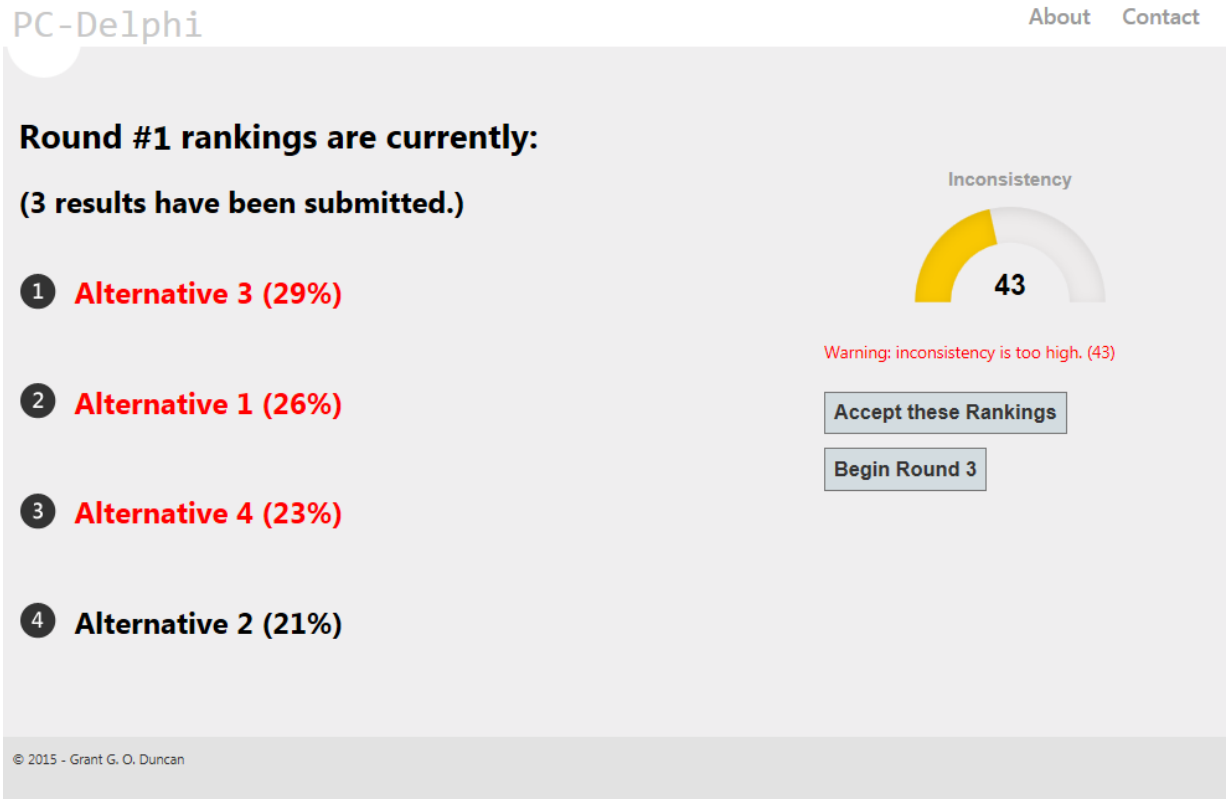

Figure 5: Example Round End Screen for administrative users

Step 5: Perform the subsequent rating rounds

The second (and subsequent) rounds should now be performed. All the participants are notified (either by the system via email or manually by the facilitator) and, if willing, should participate in the round within the given time constraints. Similarly to the round 1 ratings, the participants would slide the slider control's handle toward the desired alternative, subjectively gauging the weight of one alternative compared to the other. During this round however (and all subsequent rounds), the average aggregated (across the participants) position of the sliders is displayed on each of the sliders, as a means of feedback from the previous session, as well the most disputed triad is

identified. This allows the participants to make an informed decision based on the outcome of the previous round (s). Note that the information displayed is not cumulative, but rather is only from the round previous. Once finished, the results are submitted.

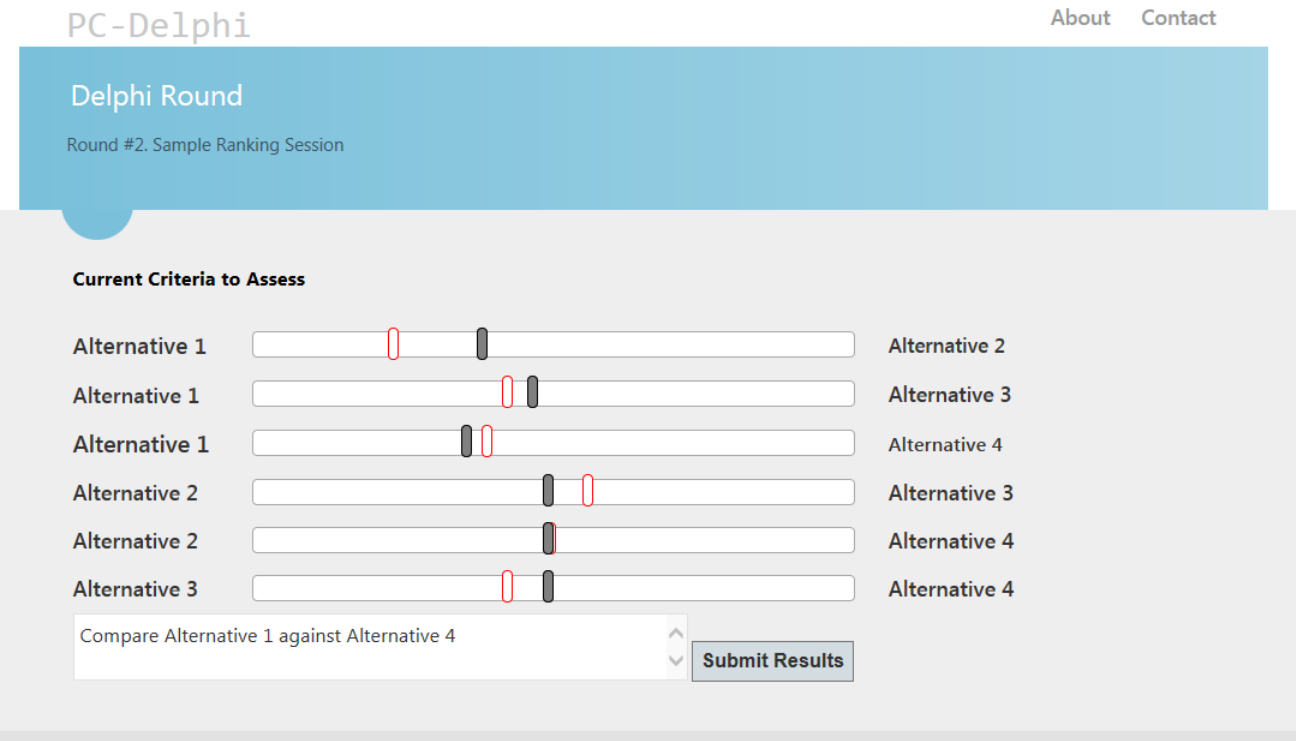

Figure 6: Second Round Rating Screen

This time, however, the aggregated rankings and inconsistency information is displayed. Participants are encouraged to submit comments to the facilitator regarding the displayed feedback. The facilitator and administrators meanwhile gauge the level of consensus and determine whether or not to go through another round. Certainly, if the system inconsistency is low, then it seems logical that the process can end. However, depending on the parameters set for the Delphi, one or more subsequent rounds may be needed. Note that the most inconsistent triad is highlighted in red in order to facilitate it's resolution (or to highlight it as a point of contention among the decision makers).

Step 6: Committing the results

Once an acceptable level of consensus is reached, or a predetermined round threshold has been achieved, the results must be committed and the session ended. The facilitator (or any administrative user) commits the results. From this point on, all participants to the session will be redirected to the results screen, which displays the final aggregated rankings, the item's associated weights, and the overall inconsistency.

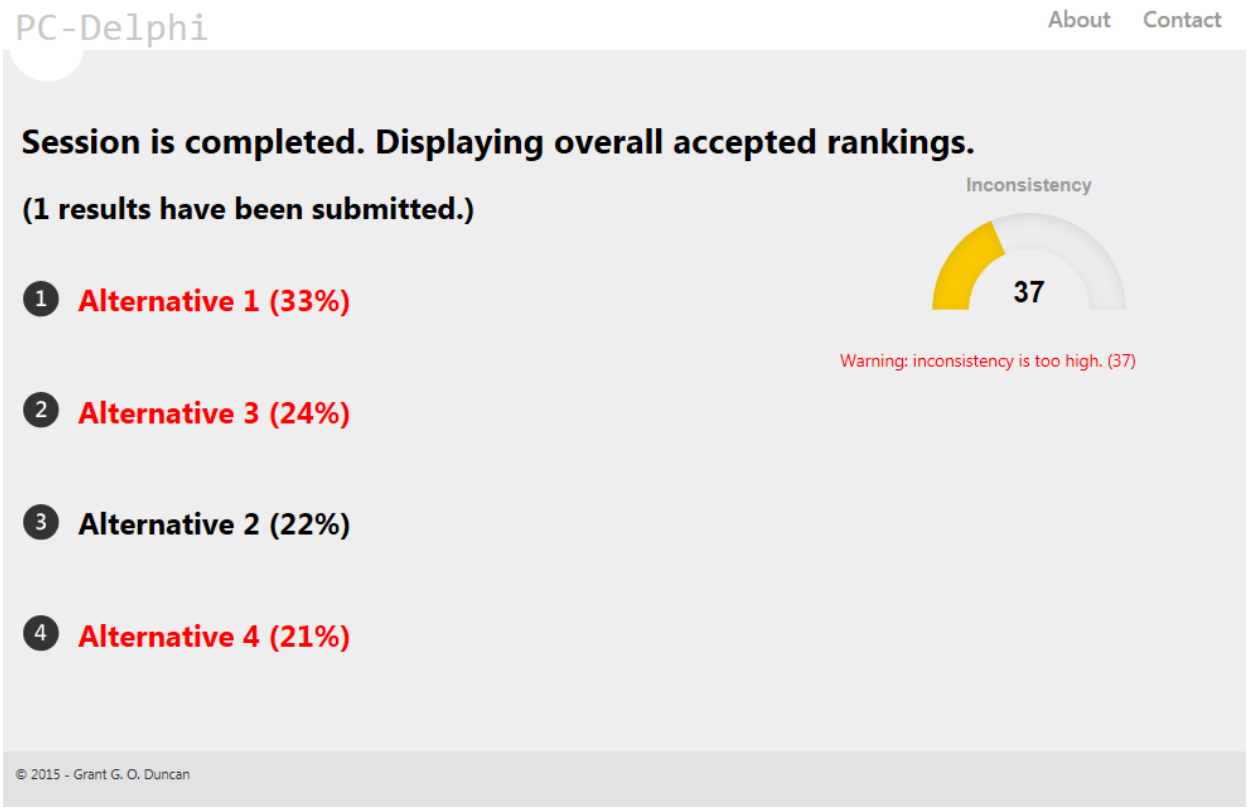

Figure 7: Final Results Screen

4 The PC Delphi System - An ASP.NET Application

The PC Delphi System is an ASP.NET web application that allows a set of users to rank a number of alternatives via pairwise comparisons over the course of 2 or more Delphi-style rounds. Utilizing a continuous scale implemented with a number of sliders, participants are asked to weigh each alternative against the other, in pairs, by moving the slider control towards the alternative that represents *more* of the attribute being compared. As the slider moves towards the heavier alternative (in this instance *heavier* is used to represent the alternative that is given more *weight*), the alternative's short description's text increases in size proportional to the distance of the slider to the alternative's label; providing a visual mnemonic which aids the participant in their assessments (called a *word map* or a *word cloud*).

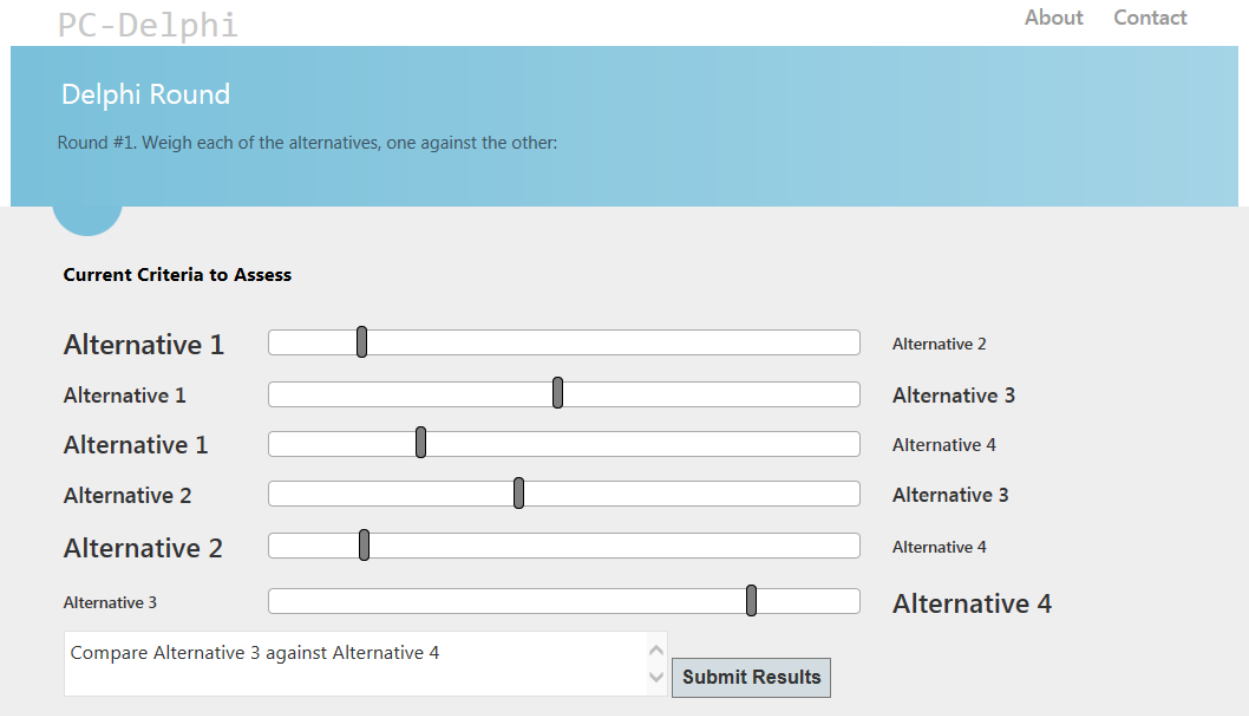

Figure 8: User Interface for round 1 of a 4 alternative PC Delphi session

Figure 1 displays the user interface of the first round of a four alternative PC Delphi session. Notice the increasing and decreasing text size that represents the degree to which one alternative has more weight than the other. The text box underneath the assessment sliders area gives the

alternatives' description (in order to better differentiate between them). Once the user is satisfied with their assessments, they click the "Submit Results" button in order to proceed to the next screen. Depending on the user's rights and their position within the pc-delphi process, the user will see either a summary screen displaying the aggregated results and other metrics associated with the process, or an "accept" screen that informs the user that their role in the current round of the process has been completed. The following sections describe the design and implementation of the PC Delphi system.

4.1 Design

4.1.1 ASP.NET and C#

The PC Delphi system is an ASP.NET web application. The ASP.NET framework (often called the .NET platform) is an "open source server-side Web application framework designed for Web development to produce dynamic Web pages" [37]. First release in 2002 as version 1.0, ASP.NET is the successor to Microsoft's ASP (Active Server Pages) technology. A memory managed virtual machine-based technology similar to Java, ASP.NET runs on the Microsoft Common Language Runtime (CLR) bytecode engine. Microsoft provides a number of programming languages that interact with the ASP.NET framework: VB.NET, Visual C++.NET, C# and F#. In theory, any programming language that can be compiled to the CLR can leverage the .NET framework. The .NET framework provides a very large class library to developers, known as the Framework Class Library, or FCL.

ASP.NET applications are tiered; the visual layer defines the user interface via a XML markup language (based on ASP) interlaced with HTML, JavaScript and CSS (and later rendered to HTML for use on the web), called the template page, though colloquially it tends to be referred to as the "ASPX page" due to the ".aspx" file extension of template source files. The logical or control layer is implemented in a .NET compatible language, and is usually called the "code behind" file. Some .NET code and expressions can be embedded into the ASP page, though this method of implementation is unmanageable for any non-trivial applications. The ASP.NET framework

provides a number of important application features: session state (since the HTTP (web) protocol is stateless), security and authentication and a common set of web-enabled interface objects, among others.

None-Microsoft .NET implementations exist, for instance, the mono project (http://www.mono-project.com/) which implements are subset of the .NET framework 4.5 and allows for the development of .NET applications running in non-Windows environments. Most .NET development, however, is done with the so-called Microsoft stack: Windows operating system, Microsoft .NET framework, Visual Studio (an IDE for the development of .NET applications), Microsoft SQL Server (as the data layer/repository) and Internet Information Services (IIS) as the web server. The PC Delphi system was developed with the Microsoft stack in mind, and as such it should be noted that compatibility with other environments may not be feasible.

4.1.2 HTML, JavaScript and CSS

Being a web application, the PC Delphi system implements it's user interface as a collection of HyperText Markup Language (HTML), JavaScript and Cascading Style Sheet (CSS) files. ASP.NET code is translated into HTML in order to represent the visual user interface in a web browser. These user interface elements are controlled by JavaScript - an interpreted dynamically typed programming language that is used to interact with the web browser client (often referred to as client-side coding). The UI is styled in CSS, a set of hierarchical definitions that style and theme a browser's user interface elements. Styles can be applied dynamically, and in response to common web browser events (like a mouse hover).

It should be noted that the development of complex web applications is often hindered by the fractured nature of web "standards". Differences in how vendors choose to implement certain features (or even if they implement certain features) can cause interoperability issues between browsers as well as unexpected behavior. This is somewhat mitigated via the use of JavaScript libraries (like the populate JQuery libraries), which present a unified interface to the programmer while handling the platform differences transparently.

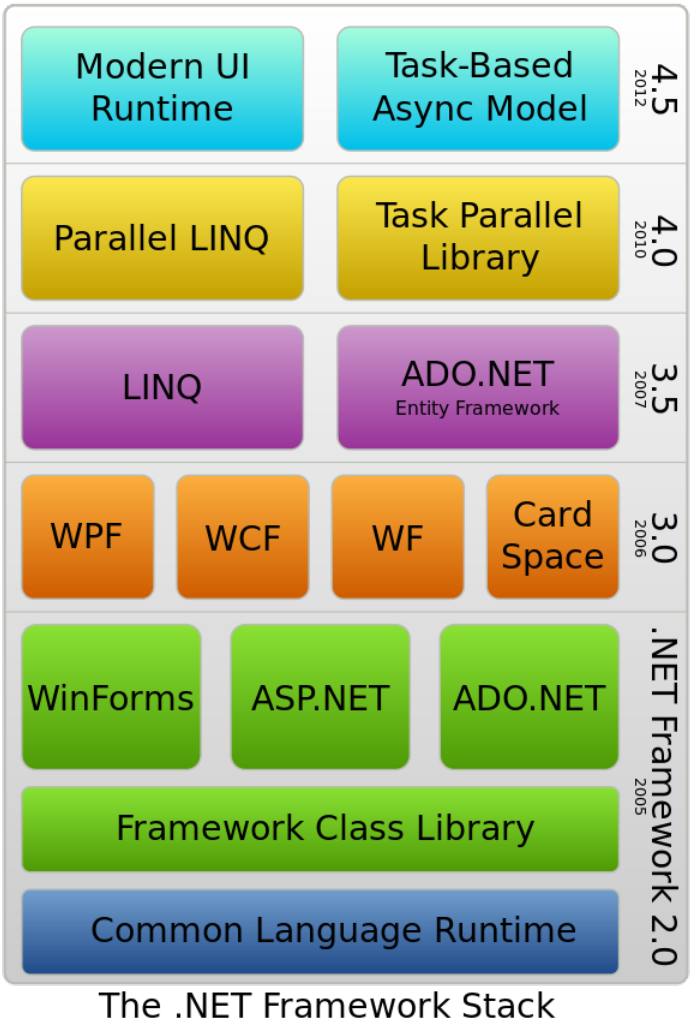

Figure 9: The .NET Framework
(retrieved under permission from http://commons.wikimedia.org/wiki/File:DotNet.svg)

4.1.3 Microsoft SQL Server

The backend data store of the reference implementation is Microsoft SQL Server, commonly referenced as SQL Server, a relational database management system developed by Microsoft. A large number of versions exist, targeting various audiences and workloads, though generally code is compatible between versions (Microsoft also supports the ability to restore backups from a higher version to three versions down). SQL Server is high performance, and supports tight integration with the .NET platform by allowing functions, procedures, triggers and constraints to be written in any CLR-compiled language. Natively, SQL Server supports Microsoft's and Sybase's extension to ANSI SQL, called Transact-SQL (TSQL), which supports procedural programming, local variables and a large library of support functions, among other changes. Communication between the reference implementation's data layer (written in C#) and the backend data store are implemented in TSQL.

4.1.4 Application Development Model

The PC Delphi system was designed as a three-tiered application, using the Model-view-controller (MVC) architectural pattern. Wherein the model encapsulates the behavior (data, logic and rules) of the application independently of the user interface (the view) [38]. Additionally the controller coordinates the interactions between the model and the view, issuing commands to manage the state of the application. The Delphi.Data and Delphi SQL DB projects comprise the model layer, while the Delphi.Web project comprises the visual and controller layers. Additionally, the project PC provides the mathematical subroutines (see Appendix [Source_code] for more information).

The application state and model are stored in a Structured Query Language (SQL) database, referenced via the execution of Transact-SQL (TSQL) stored procedures (subroutines written in Transact-SQL that are stored within the database and mediate access between an application and the database's views and tables). These stored procedures are accessed from the Delphi.Data.SQL.SqlClient class, which maps the SQL-level tables and state information into C# classes in the Delphi.Data.Entities namespace. The visual layer comprises a number of ASPX

pages and utilizes JQuery and JQuery UI, two open source (MIT licensed) JavaScript libraries, in order to simplify the development of the user interface (and ensure it is compatible with the greatest number of web browsers).

4.2 Notable .NET Features Leveraged by the Reference Implementation

4.2.1 Session Handling

Sessions provide a means by which an ASP.NET application can store persistent information, retrievable between HTTP POST actions. Implementation wise, the first request to an ASP.NET application creates a session ID, a token by which the session can be later retrieved. This ID is stored by the browser either in a cookie, or by appending the session ID to all subsequent page requests. The session is an object bag within which any .NET object may be stored. See Appendix A, Listing 1, lines 19 to 35 for an example of typical session usage.

4.2.2 Data Binding

Data binding allows an ASP.NET control to be bound to a property of the associated template's class. Utilizing a specialized binding syntax, .NET compliant programming language code is inserted between <%# and %> tags. Typically, data binding is used to create enumerated or repeated controls. Data binding can greatly simplify the implementation of a complex ASP.NET control. The reference implementation's Round.aspx template file demonstrates a good example of the use of data binding. See Appendix A, Listing 15, lines 23 to 46. This particular example demonstrates the use of a common data bound ASP.NET control, the DataList.

4.2.3 Serialization

Serialization is used to encode objects to a format compatible with storage or streaming. Typically used with XML, binary and string serialization, in theory any object that is made up of known types can be serialized. The PC Delphi reference implementation uses streaming in order to decode and

cast JSON data sent from the sliders to the application for processing and storage into the SQL database. Appendix A, Listing 24 demonstrates a typical serialization example.

4.2.4 LINQ

LINQ stands for Language Integrated Query. Introduced into the .NET framework at version 3, LINQ allows the developer to query collections of objects with a SQL-like syntax. After compilation, LINQ expressions are translated into iterators, which then enumerate through the LINQed objects at run-time. The advantages to using LINQ include lazy evaluation as well as a means to simplify the expression of complex searches and looping through collections of objects. Appendix A, Listing 1, lines 160 to 164 demonstrates typical LINQ usage.

4.3 Data Model

The application's data model is comprised of a number of entities. Figure 2 displays the entity relationship diagram that describes the model.

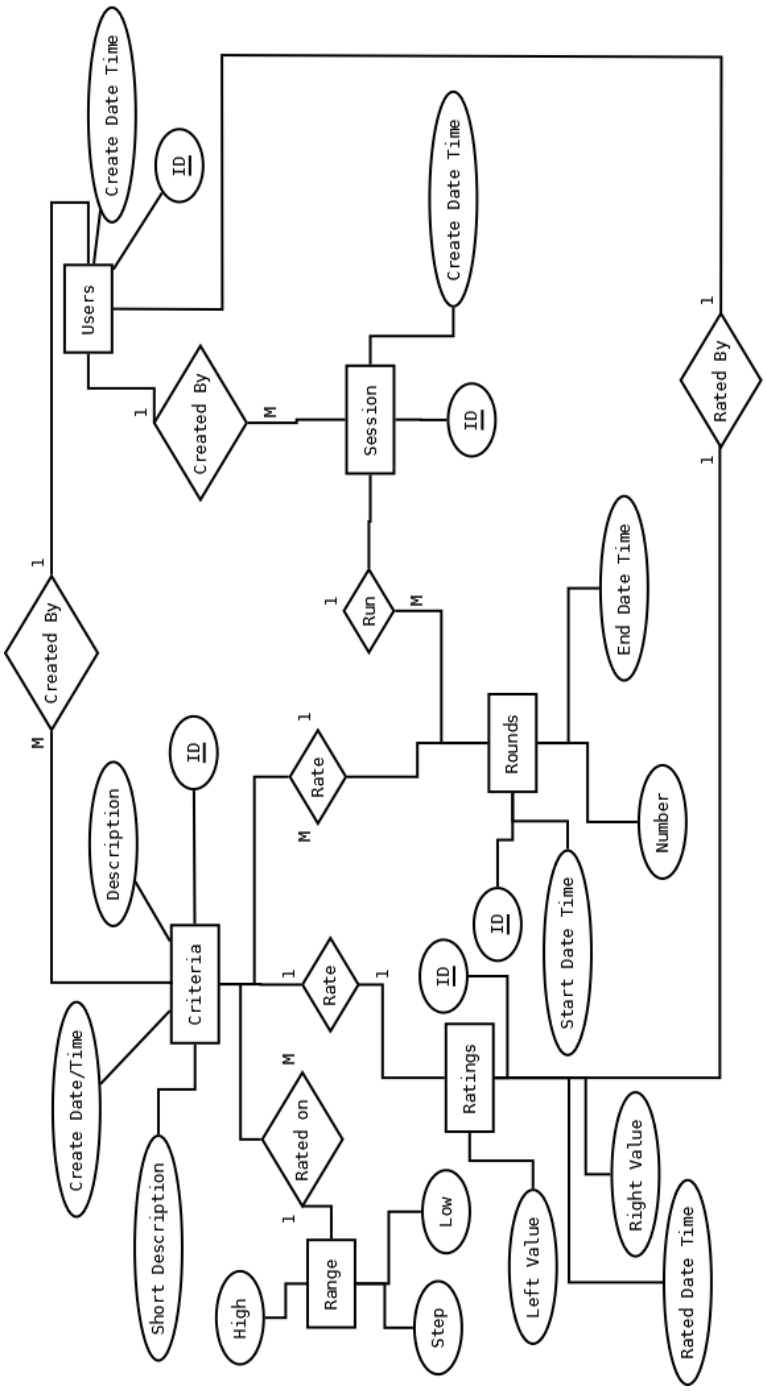

Figure 10: The PC Delphi System ERD

In descriptive terms: Users give Criteria Ratings that are rated according to specific Ranges over the course of multiple Rounds in a particular Session.

4.3.1 Entity Descriptions

The following section describes the entities that comprise the PC Delphi system's data model. The C# source code representing each of these can be found in Appendix *A*; the TSQL definitions for each can be found in Appendix *B*.

The Users Entity

The Users entity represents a specific user within the system. Each user is accessed via their globally unique identifier (or GUID). One or more users are the administrators of the session, and only those particular users are permitted to control the flow of the pc-delphi process (by either starting a new round or by ending the session and "confirming" the results). Note that typically only a single user is marked as the administrator.

The Criteria Entity

The Criteria entity represents the concepts that are compared over the course of the pc-delphi process. They describe the stimuli that are compared and are associated to a particular range.

The Range Entity

The Range entity represents the scale over which a particular criteria entity is measured. Note that the Delphi PC system is designed to function over a continuous scale and as such the Range entity does not currently support the use of discrete or integer ranges.

The Rounds Entity

The Rounds entity represents a particular Delphi round. Over the course of a round, all the participating users rate the criteria. The end of the round (and possible beginning of the new round) is decided by the session administrators.

The Sessions Entity

The Sessions entity comprises all the rounds, users and criteria of a pc-delphi session. Note that rounds and users are distinct per session, while criteria and ranges can be shared between any number of sessions.

4.4 The User Interface

4.4.1 Sliders

The PC Delphi System's user interface is built around the concept of a slider with a position, S, that is moved between two elements, $E_1$and E_2. The set $\{E_1, S, E_2\}$ will be referred to as a *slider set*. Moving the slider towards one of the elements, for instance E_1 would imply that E_1 possesses more of some quality (more *weight*) than E_2. The bulk of the user interface consists of repeated *slider sets*, representing the pairwise comparisons between each of the criteria. Note that E_1always represents the left hand side element while E_2 represents the right hand side element. The reciprocal of the comparison is not assessed by this system.

The sliders are implemented as JQuery UI slider controls, on an upwards range with a span from 1 to 999, with a default value set to 500 (giving each slider a 499 point scale). JQuery UI sliders use integer scales. In order to translate the slider position to each element's assessed values, Ev_1 and Ev_2, the following conversions are used:

$$Ev_1 = 1000 - S \tag{9}$$

$$Ev_2 = S \tag{10}$$

Thus the pairwise comparison (PC) between E_1 and E_2 is:

$$PC = \frac{1000 - S}{S} \tag{11}$$

The size of each element E_1 and E_2 is scaled according to $log(PC)$ or $log(\frac{1}{PC})$ respectively. That is, as the slider approaches a certain element, it's size increases according to the log of comparison, creating a *word map*. The scaling of the element is done in order to emphasize the weight of the assessment attributed to it. The logarithmic scaling was selected due to the Weber-Fechner law, which has established that the relationship between stimulus and perception is logarithmic.

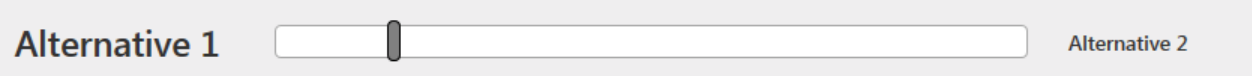

Figure 11: A typical *slider set*

From figure 3 it is apparent that as the slider thumb moves towards the element "Alternative 1", it's size increases. Subsequent rounds add an additional user interface element, the average position of the slider as calculated by all submissions from the previous round. This gives the user additional feedback as to selections of the group as a whole, as is required by the pc-delphi process. Figure 4 displays a *slider set* and the average position indicator (in red).

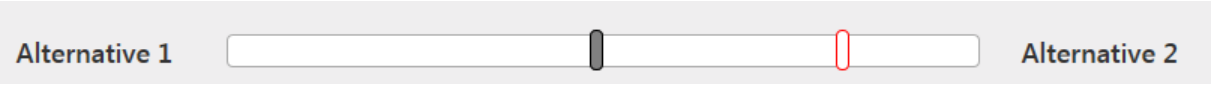

Figure 12: A *slider set* with the previous round's average position indicated

4.4.2 The Results Screen

The results screen displays the currently calculated ranking of all the alternatives and their weight as a percentage, the number of submitted assessments and the system's global inconsistency (displayed on a scale). The session facilitator, in consultation with the other participants, is also given the ability to either "Accept" the rankings and end the session, or to begin a new round. Figure 5 displays the administrator's view of the results screen after Round 1 (with two users having submitted results). A warning indicator is displayed if the inconsistency is greater than a given threshold (by default greater than $\frac{1}{3}$). While typically these results are marked as outliers and removed, it is possible that all currently submitted values fall into the outlier range; the system displays these in order to provide feedback rather than providing no results.

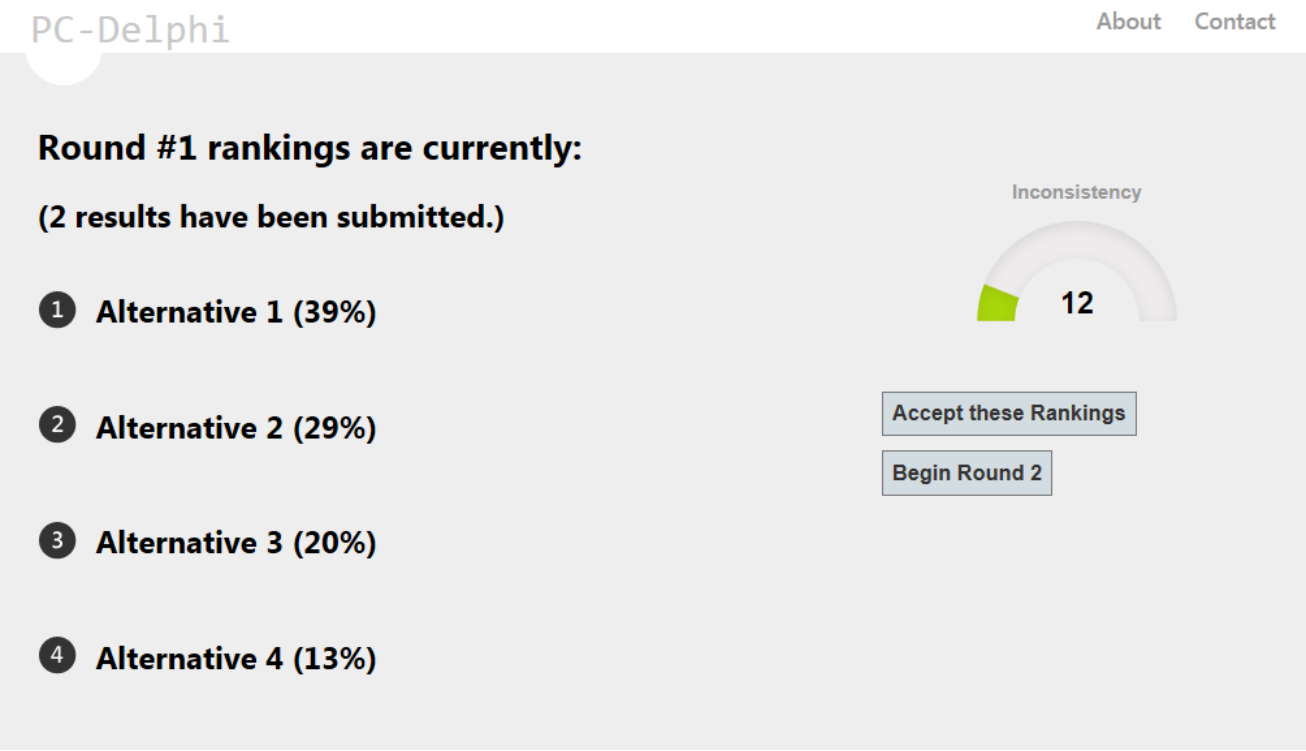

Figure 13: The administrator's view of the Results screen

4.5 Calculating the Rankings

The Results screen rankings are calculated via a multi-step process. In the first step, each individual's results are placed into a PC matrix, and their inconsistency indicator is calculated. If this value is greater than the $\frac{1}{3}$ threshold, they are removed (as these are considered outliers). In the case that all the submissions are outliers, then no outlier entries are removed and all entries contribute to the overall rankings (additionally, the Results screen will display a warning about the inconsistency being too high). In the second step, the acceptable entries are accumulated into a PC matrix, and the inconsistency is calculated from this resultant PC system. The weights of the rankings are calculated as the normalized vector of geometric means of the PC system.

4.5.1 Creating a PC matrix from the submitted ratings

The first step of calculating the rankings is to transform the submitted rating data into a PC matrix form. Recall from formula 3 that the PC comparison between two elements, E_1 and E_2, in a given a *slider set* is equal to $\frac{1000-S}{S}$ (with S representing the current value of the slider). The following algorithm details the process:

Algorithm 1 Creating a PC matrix from a set of slider sets

```
S = Set of Slider Sets position values
N = Length of S NumComps = (N * (N-1))/2
PC = matrix of size NumComps x NumComps
SIndex = 0
for i = 0 to N
  for j = 0 to N
    if i == j
      PC(i, j) = 1
    else
      if i < j PC[i, j] = (1000 - S[SIndex])/S[SIndex]
      else if i > j PC[i, j] = S[SIndex]/(1000-S[SIndex])
      increment SIndex
    end else
  end for
end for
```

Note that from the above algorithm the resultant PC matrix is reciprocal, since, as mentioned, the PC Delphi system only tests the comparison $\frac{E_1}{E_2}$ and not $\frac{E_2}{E_1}$.

4.6 Application Layout and Flow

The PC Delphi System is a web application composed of multiple web application pages (ASPX pages). The index page is called "Consent.aspx", and displays some consent notices. Once a user has consented to using the system, these messages will no longer be displayed. Users are differentiated by their Globally Unique Identifier values (GUID), which is passed as a parameter to the Consent.aspx page. Upon connection, a user context is retrieved from the database. This user context consists of a $\{Round, Session, User\}$ tuple. Note that this context tuple is persisted within the application's session state; thus it does not need to be requeried for subsequent page

transfers or refreshes (in a given browser session). Figure 6 displays the logical application flow for a PC-Delphi session.

4.6.1 Consent.aspx

The Consent.aspx module is the first page that any user attempting to use the system must first access. It has several purposes: to ensure that users have consented to the terms and conditions of the application before usage, to create the user context tuple and store it into the application's session state and to redirect the accessing user to the proper step within the current session. In the case where a user GUID does not exist, a redirect to an error message will occur and their browser session will end. Otherwise, the system will verify whether or not the accessing user has submitted their assessments for the current round. If not, they are redirected to the Rounds.aspx module in order to input their assessment ratings, if so, they will be redirected to the Results.aspx module.

4.6.2 Rounds.aspx

The Rounds.aspx module handles the extract, transform, load and store of a pc-delphi session's assessment information. Upon entry, the module retrieves the criteria (and their average assessments from the previous round) from the database, and databinds them to a templated data list control. When the user has finished their assessments, upon clicking the "Submit" button, the assessment information is stored in a JSON (JavaScript Object Notation) array and submitted via an HTTP POST. On the server-end, this information is deserialized into a list of Ratings objects, which are then sent to the data access layer for storage in the backing SQL database.

4.6.3 Results.aspx

The Results.aspx module is used to display the collated results of the current round of a pc-delphi session, or in the case of a completed session, the final round results. The results consist of a ranking of the alternatives, their calculated weights, and the system's inconsistency.

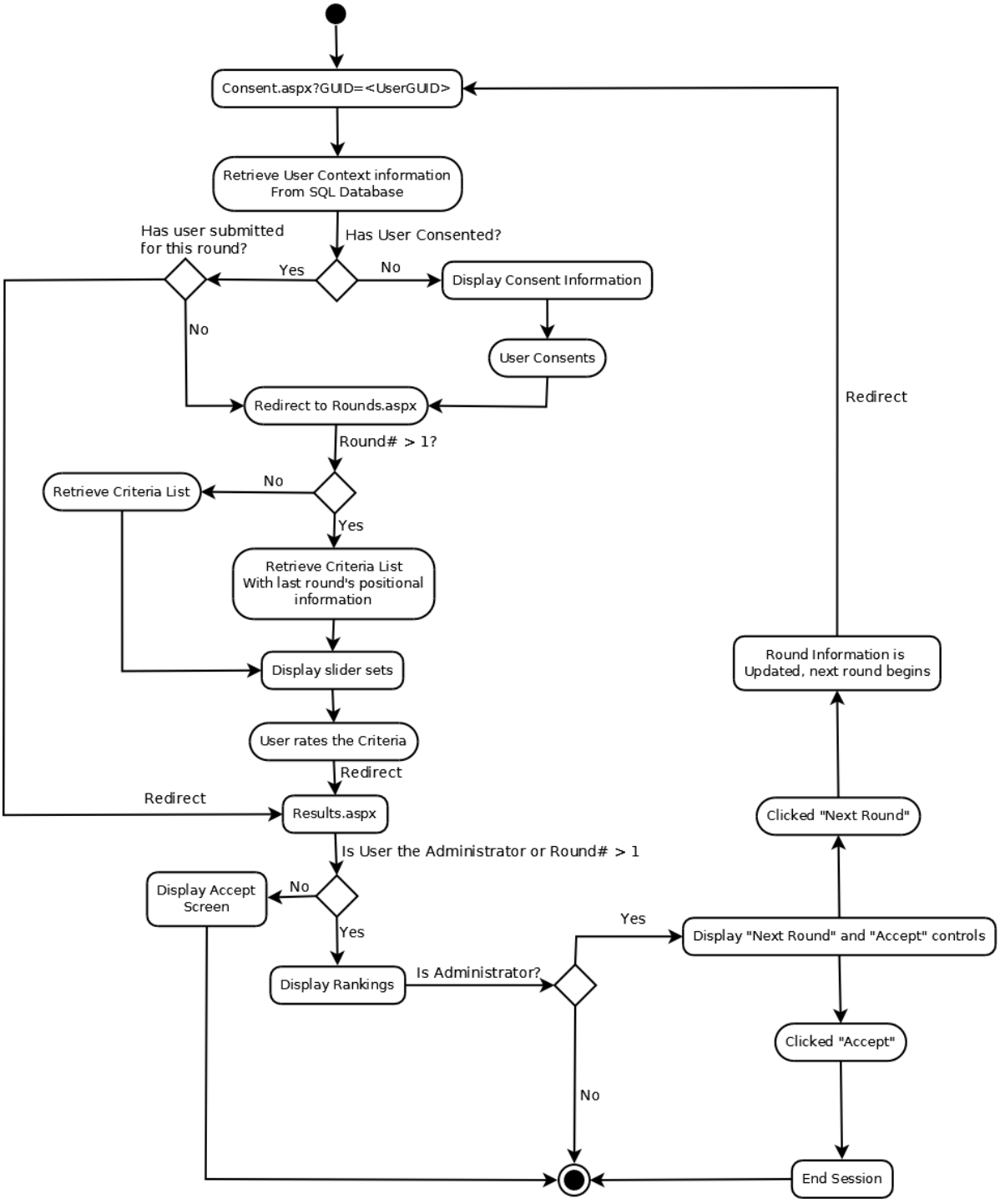

Figure 14: Top level application flowchart. Note that error states are suppressed in order to simplify the display.

5 Conclusions

The PC Delphi combines the internet, the Delphi method and the method of pairwise comparisons to create a reliable ranking system that has broad applications. The use of Koczkodaj's inconsistency indicator to assess consensus and remove outliers represent a novel and unique approach to the problem. The included reference implementation provides an easy to use and easy to deploy solution freely available to all who want to perform a PC Delphi. The reference code can also be used to guide prospective developers with their own implementation by demonstrating an easy to adapt slider widget created with the cross platform JQuery library. Given a large enough server hosting the application, this system may even be used for a referendum in a sizable country (given that it is based on Int32 values, it should in theory support over 2 billion users).

References

[1] W. W. Koczkodaj, R. Szwarc, On Axiomatization of Inconsistency Indicators for Pairwise Comparisons, Fundamenta Informaticae (4) (2014) 485–500. doi:10.3233/FI-2014-1055.

[2] M. W. Herman, W. W. Koczkodaj, A monte carlo study of pairwise comparison, Information Processing Letters 57 (1) (1996) 25–29.

[3] K. W. W. Fulop, J., S. J. Szarek, A different perspective on a scale for pairwise comparisons, in: D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, N. T. Nguyen, R. Kowalczyk (Eds.), Transactions on Computational Collective Intelligence I, Vol. 6220, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 71–84.

[4] Ramon llull, page Version ID: 635224170 (Dec. 2014).

[5] S. Bozoki, T. Rapcsak, On saaty's and koczkodaj's inconsistencies of pairwise comparison matrices, Journal of Global Optimization 42 (2) (2008) 157–175. doi:10.1007/s10898-007-9236-z.
URL http://link.springer.com/10.1007/s10898-007-9236-z

[6] W. W. Koczkodaj, S. J. Szarek, On distance-based inconsistency reduction algorithms for pairwise comparisons., Logic Journal of the IGPL 18 (6) (2010) 859–869.

[7] J. Pelaez, M. Lamata, A new measure of consistency for positive reciprocal matrices, Computers & Mathematics with Applications 46 (12) (2003) 1839–1845. doi:10.1016/S0898-1221(03)90240-9.
URL http://linkinghub.elsevier.com/retrieve/pii/S0898122103902409

[8] W. W. Koczkodaj, A new definition of consistency of pairwise comparisons, Mathematical and Computer Modelling 18 (7) (1993) 79–84. doi:10.1016/0895-7177(93)90059-8.
URL http://linkinghub.elsevier.com/retrieve/pii/0895717793900598

[9] Z. Duszak, W. W. Koczkodaj, Generalization of a new definition of consistency for pairwise comparisons, Information Processing Letters 52 (5) (1994) 273–276. doi:10.1016/0020-0190(94)00155-3.
URL http://linkinghub.elsevier.com/retrieve/pii/0020019094001553

[10] W. W. Koczkodaj, M. W. Herman, M. Orlowski, Using consistency-driven pairwise comparisons in knowledge-based systems, ACM Press, 1997, pp. 91–96. doi:10.1145/266714.266867.
URL http://portal.acm.org/citation.cfm?doid=266714.266867

[11] R. D. Luce, W. Edwards, The derivation of subjective scales from just noticeable differences, Psychological Review 65 (4) (1958) 222–237.

[12] G. Crawford, C. Williams, A note on the analysis of subjective judgment matrices, Journal of Mathematical Psychology 29 (4) (1985) 387–405. doi:10.1016/0022-2496(85)90002-1.
URL http://linkinghub.elsevier.com/retrieve/pii/0022249685900021

[13] S. Dehaene, The neural basis of the weber–fechner law: a logarithmic mental number line, Trends in cognitive sciences 7 (4) (2003) 145–147.

[14] T. L. Saaty, Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process, Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales. Serie A. Matematicas 102 (2) (2008) 251–318. doi:10.1007/BF03191825.
URL http://link.springer.com/10.1007/BF03191825

[15] Weber-fechner law, page Version ID: 634449783 (Dec. 2014).

[16] C. Okoli, S. D. Pawlowski, The delphi method as a research tool: an example, design considerations and applications, Information & Management 42 (1) (2004) 15–29. doi:10.1016/j.im.2003.11.002.
URL http://linkinghub.elsevier.com/retrieve/pii/S0378720603001794

[17] N. Dalkey, O. Helmer, An experimental application of the delphi method to the use of experts, Management science 9 (3) (1963) 458–467.

[18] H. A. Linstone, M. Turoff, The Delphi method: techniques and applications, Addison-Wesley Pub. Co., Advanced Book Program, Reading, Mass., 1975.

[19] G. Rowe, G. Wright, F. Bolger, Delphi: a reevaluation of research and theory, Technological Forecasting and Social Change 39 (3) (1991) 235–251.

[20] R. C. Schmidt, Managing delphi surveys using nonparametric statistical techniques, Decision Sciences 28 (3) (1997) 763–774. doi:10.1111/j.1540-5915.1997.tb01330.x.
URL http://doi.wiley.com/10.1111/j.1540-5915.1997.tb01330.x

[21] R. D. Needham, R. C. Loe, THE POLICY DELPHI: PURPOSE, STRUCTURE, AND APPLICATION, The Canadian Geographer/Le Geographe canadien 34 (2) (1990) 133–142. doi:10.1111/j.1541-0064.1990.tb01258.x.
URL http://doi.wiley.com/10.1111/j.1541-0064.1990.tb01258.x

[22] M. I. Yousuf, Using experts opinions through delphi technique, Practical Assessment, Research & Evaluation 12 (4) (2007) 1–8.

[23] J. SANDERS, B. WORTHEN, EDUCATIONAL-EVALUATION - ALTERNATIVE APPROACHES AND PRACTICAL GUIDELINES - AN ALTERNATE PERSPECTIVE - RESPONSE, EVALUATION AND PROGRAM PLANNING 15 (3) (1992) 340.

[24] R. Loo, The delphi method: a powerful tool for strategic management, Policing: An International Journal of Police Strategies & Management 25 (4) (2002) 762–769. doi:10.1108/13639510210450677.
URL http://www.emeraldinsight.com/doi/abs/10.1108/13639510210450677

[25] K. J. Arrow, A difficulty in the concept of social welfare, The Journal of Political Economy (1950) 328–346.

[26] F. T. Hartman, A. Baldwin, Using technology to improve delphi method., Journal of Computing in Civil Engineering 9 (4) (1995) 244.

[27] M. Turoff, S. R. Hiltz, COMPUTER BASED DELPHI PROCESSES.
URL http://web.njit.edu/ turoff/Papers/delphi3.html

[28] X. Yang, L. Yan, L. Zeng, How to handle uncertainties in AHP: The cloud delphi hierarchical analysis, Information Sciences 222 (2013) 384–404. doi:10.1016/j.ins.2012.08.019.
URL http://linkinghub.elsevier.com/retrieve/pii/S0020025512005725

[29] C.-M. Wu, C.-L. Hsieh, K.-L. Chang, A hybrid multiple criteria decision making model for supplier selection, Mathematical Problems in Engineering 2013 (2013) 1–8. doi:10.1155/2013/324283.
URL http://www.hindawi.com/journals/mpe/2013/324283/

[30] K. I. Vatalis, D. C. Kaliampakos, An overall index of environmental quality in coal mining areas and energy facilities, Environmental Management 38 (6) (2006) 1031–1045. doi:10.1007/s00267-005-0114-5.
URL http://link.springer.com/10.1007/s00267-005-0114-5

[31] I. Lipuscek, M. Bohanec, L. Oblak, L. Zadnik Stirn, A multi-criteria decision-making model for classifying wood products with respect to their impact on environment, The International Journal of Life Cycle Assessment 15 (4) (2010) 359–367. doi:10.1007/s11367-010-0157-6.
URL http://link.springer.com/10.1007/s11367-010-0157-6

[32] D. F. Kocaoglu, Portland International Center for Management of Engineering and Technology, PICMET, Portland International Conference on Management of Engineering & Technology, Portland International Conference on Management of Engineering & Technology, 2008: PICMET 2008 ; Cape Town, South Africa, 27 - 31 July 2008, IEEE Service Center, Piscataway, NJ, 2008.
URL http://ieeexplore.ieee.org/servlet/opac?punumber=4591409

[33] G. A. Mendoza, R. Prabhu, Development of a methodology for selecting criteria and indicators of sustainable forest management: A case study on participatory assessment, Environmental Management 26 (6) (2000) 659–673. doi:10.1007/s002670010123.
URL http://link.springer.com/10.1007/s002670010123

[34] S. Bozóki, J. Fülöp, W. W. Koczkodaj, An lp-based inconsistency monitoring of pairwise comparison matrices, Mathematical and Computer Modelling 54 (1) (2011) 789–793.

[35] P. Legendre, Species associations: the kendall coefficient of concordance revisited, Journal of Agricultural, Biological, and Environmental Statistics 10 (2) (2005) 226–245.

[36] E. A. Holey, J. L. Feeley, J. Dixon, V. J. Whittaker, An exploration of the use of simple statistics to measure consensus and stability in delphi studies, BMC Medical Research Methodology 7 (1) (2007) 52. doi:10.1186/1471-2288-7-52.
URL http://www.biomedcentral.com/1471-2288/7/52

[37] ASP.NET - wikipedia, the free encyclopedia.
URL http://en.wikipedia.org/wiki/ASP.NET

[38] Model-view-controller - wikipedia, the free encyclopedia.
URL http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[39] H. A. Linstone, M. Turoff, et al., The Delphi method: Techniques and applications, Vol. 29, Addison-Wesley Reading, MA, 1975.

[40] J. Liebowitz, Useful approach for evaluating expert systems, Expert Systems 3 (2) (1986) 86–96. doi:10.1111/j.1468-0394.1986.tb00198.x.
URL http://doi.wiley.com/10.1111/j.1468-0394.1986.tb00198.x

[41] F. Casati, S. Ceri, B. Pernici, G. Pozzi, Deriving active rules for workflow enactment, in: G. Goos, J. Hartmanis, J. van Leeuwen, R. R. Wagner, H. Thoma (Eds.), Database and Expert Systems Applications, Vol. 1134, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, pp.

94–115.

URL http://link.springer.com/10.1007/BFb0034673

[42] F. Casati, S. Castano, M. Fugini, I. Mirbel, B. Pernici, Using patterns to design rules in workflows, IEEE Transactions on Software Engineering 26 (8) (2000) 760–785. doi:10.1109/32.879813.

URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=879813

A PC Delphi ASP.NET, C# and JavaScript Source Code

A.1 Delphi.Web Source Code Listing

The following source code listing (27 files in total) comprises the ASP.NET or Web portion of the PC Delphi implementation. Note that auto-generated and default files (for instance, JQuery source files) are not included in this listing. This web application is designed to be deployed to a compliant ASP.NET web server, running the .NET framework 4.5 or higher.

Listing 1: Round.aspx.cs

```
1  //
2  // Rounds.aspx.cs
3  // (c) 2014 Grant G. O. Duncan <gg_duncan@laurentian.ca>
4  // Code behind file for the Round.aspx webform.
5  // This page runs a round for the given session.
6  //
7  using System;
8  using System.Collections.Generic;
9  using System.Linq;
10 using System.Web;
11 using System.Web.UI;
12 using System.Web.UI.WebControls;
13 using Delphi.Data.Entities;
14
15 namespace Delphi.Web
16 {
17   public partial class Round : System.Web.UI.Page
18   {
19     private Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
           DelphiSession, Delphi.Data.Entities.User> _SesInfo = null;
```

```
20      protected Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
            DelphiSession, Delphi.Data.Entities.User> SesInfo
21      {
22        get
23        {
24          if (_SesInfo == null)
25          {
26            _SesInfo = Session["SessionInfo"] as Tuple<Delphi.Data.
                  Entities.Round, Delphi.Data.Entities.DelphiSession, Delphi
                  .Data.Entities.User>;
27          }
28          return _SesInfo;
29        }
30        set
31        {
32          Session["SessionInfo"] = value;
33          _SesInfo = value;
34        }
35      }
36
37      private Delphi.Data.SQL.SqlClient _SQL = null;
38      private Delphi.Data.SQL.SqlClient SQL
39      {
40        get
41        {
42          if (_SQL == null)
43          {
44            _SQL = Session["SqlClient"] as Delphi.Data.SQL.SqlClient;
45          }
```

```
46          return _SQL;
47        }
48        set
49        {
50          Session["SqlClient"] = value;
51          _SQL = value;
52        }
53      }
54
55      protected int RoundNumber { get { return SesInfo.Item1.Round_Number
            ; } }
56      private List<Tuple<Criteria, Criteria>> _CriteriaList = null;
57      protected List<Tuple<Criteria, Criteria>> CriteriaList
58      {
59        get
60        {
61          if (_CriteriaList == null)
62          {
63            _CriteriaList = Session["CriteriaList"] as List<Tuple<
                Criteria, Criteria>>;
64          }
65          return _CriteriaList;
66        }
67        private set
68        {
69          Session["CriteriaList"] = value;
70          _CriteriaList = value;
71        }
72      }
```

```
73
74      private List<Tuple<Criteria, Criteria, string>>
            _CriteriaListWithPosition = null;
75      protected List<Tuple<Criteria, Criteria,string>>
            CriteriaListWithPosition
76      {
77        get
78        {
79          if (_CriteriaListWithPosition == null)
80          {
81            _CriteriaListWithPosition = Session["CriteriaListWithPosition
                "] as List<Tuple<Criteria, Criteria,string>>;
82          }
83          return _CriteriaListWithPosition;
84        }
85        private set
86        {
87          Session["CriteriaListWithPosition"] = value;
88          _CriteriaListWithPosition = value;
89        }
90      }
91
92      protected int IDIterator { get; set; }
93      private Delphi.Data.SQL.SqlClient SqlClient = new Delphi.Data.SQL.
            SqlClient();
94      protected Criteria[] Crits { get; private set; }
95
96      protected void Page_Load(object sender, EventArgs e)
97      {
```

```
98          if (!this.IsPostBack)
99          {
100           IDIterator = 0;
101           // Has some session info, so probably can proceed:
102           if (SesInfo != null && SesInfo.Item3.HasConsented)
103           {
104             Crits = SqlClient.GetCriteriaBySessionID(SesInfo.Item2.
                    Session_ID).ToArray();
105             CriteriaList = Criteria.Make_PC_Criteria(Crits);
106             Get_Last_Round_Position_Data();
107             RoundItems.DataSource = CriteriaListWithPosition;
108             RoundItems.DataBind();
109           }
110           else
111           {
112             Response.Redirect("Consent.aspx");
113           }
114         }
115       }
116
117       private void Get_Last_Round_Position_Data()
118       {
119         CriteriaListWithPosition = new List<Tuple<Criteria, Criteria,
                string>>();
120         List<Tuple<int, int, int>> PrevRatings = SQL.
                GetLastRoundBySessionID(SesInfo.Item2.Session_ID);
121         IEnumerable<Tuple<int, int, int>> AvgPosition = null;
122         foreach (Tuple<Criteria, Criteria> crits in CriteriaList)
123         {
```

```
124         if(PrevRatings.Count > 0)
125         {
126           AvgPosition = from Tuple<int, int, int> t in PrevRatings
127                         where crits.Item1.Criteria_ID == t.Item1 &&
128                               crits.Item2.Criteria_ID == t.Item2
129                         select t;
130           if (AvgPosition.Count() > 0)
131             CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                    string>(crits.Item1, crits.Item2, "[500," + AvgPosition.
                    First().Item3.ToString() + "]"));
132           else
133             CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                    string>(crits.Item1, crits.Item2, "[500]"));
134         }
135         else
136           CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                 string>(crits.Item1, crits.Item2, "[500]"));
137       }
138     }
139
140     protected void Btn_Submit_Click(object sender, EventArgs e)
141     {
142       try
143       {
144         IEnumerable<Delphi.Web.Data.ComparisonData> FoundResults = null
                ;
145         Delphi.Web.Data.ComparisonData Result = null;
146         List<Rating> Ratings = new List<Rating>();
147
```

```
148         // Deserialize the JSON comparison data:
149         Delphi.Web.Data.ComparisonData[] ComparisonResults = Delphi.Web
                .Data.ComparisonData.Deserialize(Comparison_Values.Value);
150         //
151         // Grab the results:
152         // Constructs a list of ratings
153         // Resolves for all criteria:criteria comparisons, using 1 for
                those items that
154         // do not exist in the submitted result set
155         //
156         foreach (Tuple<Criteria, Criteria> Comparison in CriteriaList)
157         {
158           if (ComparisonResults != null)
159           {
160             FoundResults = from Delphi.Web.Data.ComparisonData
                    ComparisonResult in ComparisonResults
161                                 where ComparisonResult != null
162                                 && ComparisonResult.lhs.id == Comparison.
                                        Item1.Criteria_ID
163                                 && ComparisonResult.rhs.id == Comparison.
                                        Item2.Criteria_ID
164                                 select ComparisonResult;
165             if (FoundResults.Count() > 0) Result = FoundResults.
                    FirstOrDefault();
166             else Result = null;
167           }
168           else Result = null;
169           if (Result != null)
170           {
```

```
171             Ratings.Add(new Rating()
172             {
173               // LHS Criteria:
174               LHS_Criteria = Comparison.Item1,
175               LHS_Criteria_Value = (decimal)Result.lhs.value,
176               // Rated by and when:
177               Rated_By = SesInfo.Item3, /* Item3 = user */
178               Rated_DtTm = DateTime.Now,
179               // RHS Criteria:
180               RHS_Criteria = Comparison.Item2,
181               RHS_Criteria_Value = (decimal)Result.rhs.value,
182               Slider_Pos = Result.lhs.sliderValue,
183               Rating_ID = -1
184             });
185           }
186           else
187           {
188             Ratings.Add(new Rating()
189             {
190               // LHS Criteria:
191               LHS_Criteria = Comparison.Item1,
192               LHS_Criteria_Value = 1m,
193               // Rated by and when:
194               Rated_By = SesInfo.Item3, /* Item3 = user */
195               Rated_DtTm = DateTime.Now,
196               // RHS Criteria:
197               RHS_Criteria = Comparison.Item2,
198               RHS_Criteria_Value = 1m,
199               Slider_Pos = 500,
```

```
200                     Rating_ID = -1
201                 });
202             }
203         }
204         // List is built, let's dump it to the DB:
205         SQL.Add_Update_Ratings(Ratings, SesInfo.Item1.Round_ID);
206       }
207       catch
208       {
209         Response.Redirect("Error.aspx");
210       }
211       // Let's go see the results:
212       if (SesInfo.Item1.Session_Is_Finished || SesInfo.Item3.
              IsAdministrator)
213         Response.Redirect("Results.aspx");
214       else
215         Response.Redirect("RoundEnd.aspx");
216     }
217   }
218 }
```

Listing 2: Round.aspx.cs

```
1  //
2  // Rounds.aspx.cs
3  // (c) 2014 Grant G. O. Duncan <gg_duncan@laurentian.ca>
4  // Code behind file for the Round.aspx webform.
5  // This page runs a round for the given session.
6  //
7  using System;
```

```
8  using System.Collections.Generic;
9  using System.Linq;
10 using System.Web;
11 using System.Web.UI;
12 using System.Web.UI.WebControls;
13 using Delphi.Data.Entities;
14
15 namespace Delphi.Web
16 {
17   public partial class Round : System.Web.UI.Page
18   {
19     private Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> _SesInfo = null;
20     protected Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> SesInfo
21     {
22       get
23       {
24         if (_SesInfo == null)
25         {
26           _SesInfo = Session["SessionInfo"] as Tuple<Delphi.Data.
                Entities.Round, Delphi.Data.Entities.DelphiSession, Delphi
                .Data.Entities.User>;
27         }
28         return _SesInfo;
29       }
30       set
31       {
32         Session["SessionInfo"] = value;
```

```
33          _SesInfo = value;
34        }
35      }
36
37      private Delphi.Data.SQL.SqlClient _SQL = null;
38      private Delphi.Data.SQL.SqlClient SQL
39      {
40        get
41        {
42          if (_SQL == null)
43          {
44            _SQL = Session["SqlClient"] as Delphi.Data.SQL.SqlClient;
45          }
46          return _SQL;
47        }
48        set
49        {
50          Session["SqlClient"] = value;
51          _SQL = value;
52        }
53      }
54
55      protected int RoundNumber { get { return SesInfo.Item1.Round_Number
            ; } }
56      private List<Tuple<Criteria, Criteria>> _CriteriaList = null;
57      protected List<Tuple<Criteria, Criteria>> CriteriaList
58      {
59        get
60        {
```

```
61          if (_CriteriaList == null)
62          {
63            _CriteriaList = Session["CriteriaList"] as List<Tuple<
                  Criteria, Criteria>>;
64          }
65          return _CriteriaList;
66        }
67        private set
68        {
69          Session["CriteriaList"] = value;
70          _CriteriaList = value;
71        }
72      }
73
74      private List<Tuple<Criteria, Criteria, string>>
           _CriteriaListWithPosition = null;
75      protected List<Tuple<Criteria, Criteria,string>>
           CriteriaListWithPosition
76      {
77        get
78        {
79          if (_CriteriaListWithPosition == null)
80          {
81            _CriteriaListWithPosition = Session["CriteriaListWithPosition
                  "] as List<Tuple<Criteria, Criteria,string>>;
82          }
83          return _CriteriaListWithPosition;
84        }
85        private set
```

```
86          {
87              Session["CriteriaListWithPosition"] = value;
88              _CriteriaListWithPosition = value;
89          }
90      }
91
92      protected int IDIterator { get; set; }
93      private Delphi.Data.SQL.SqlClient SqlClient = new Delphi.Data.SQL.
            SqlClient();
94      protected Criteria[] Crits { get; private set; }
95
96      protected void Page_Load(object sender, EventArgs e)
97      {
98          if (!this.IsPostBack)
99          {
100             IDIterator = 0;
101             // Has some session info, so probably can proceed:
102             if (SesInfo != null && SesInfo.Item3.HasConsented)
103             {
104                 Crits = SqlClient.GetCriteriaBySessionID(SesInfo.Item2.
                        Session_ID).ToArray();
105                 CriteriaList = Criteria.Make_PC_Criteria(Crits);
106                 Get_Last_Round_Position_Data();
107                 RoundItems.DataSource = CriteriaListWithPosition;
108                 RoundItems.DataBind();
109             }
110             else
111             {
112                 Response.Redirect("Consent.aspx");
```

```
113             }
114         }
115     }
116
117     private void Get_Last_Round_Position_Data()
118     {
119         CriteriaListWithPosition = new List<Tuple<Criteria, Criteria,
                string>>();
120         List<Tuple<int, int, int>> PrevRatings = SQL.
                GetLastRoundBySessionID(SesInfo.Item2.Session_ID);
121         IEnumerable<Tuple<int, int, int>> AvgPosition = null;
122         foreach (Tuple<Criteria, Criteria> crits in CriteriaList)
123         {
124             if(PrevRatings.Count > 0)
125             {
126                 AvgPosition = from Tuple<int, int, int> t in PrevRatings
127                               where crits.Item1.Criteria_ID == t.Item1 &&
128                                     crits.Item2.Criteria_ID == t.Item2
129                               select t;
130                 if (AvgPosition.Count() > 0)
131                     CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                            string>(crits.Item1, crits.Item2, "[500," + AvgPosition.
                            First().Item3.ToString() + "]"));
132                 else
133                     CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                            string>(crits.Item1, crits.Item2, "[500]"));
134             }
135             else
```

```
136             CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                    string>(crits.Item1, crits.Item2, "[500]"));
137         }
138     }
139
140     protected void Btn_Submit_Click(object sender, EventArgs e)
141     {
142         try
143         {
144             IEnumerable<Delphi.Web.Data.ComparisonData> FoundResults = null
                    ;
145             Delphi.Web.Data.ComparisonData Result = null;
146             List<Rating> Ratings = new List<Rating>();
147
148             // Deserialize the JSON comparison data:
149             Delphi.Web.Data.ComparisonData[] ComparisonResults = Delphi.Web
                    .Data.ComparisonData.Deserialize(Comparison_Values.Value);
150             //
151             // Grab the results:
152             // Constructs a list of ratings
153             // Resolves for all criteria:criteria comparisons, using 1 for
                    those items that
154             // do not exist in the submitted result set
155             //
156             foreach (Tuple<Criteria, Criteria> Comparison in CriteriaList)
157             {
158                 if (ComparisonResults != null)
159                 {
```

```
160             FoundResults = from Delphi.Web.Data.ComparisonData
                    ComparisonResult in ComparisonResults
161                             where ComparisonResult != null
162                             && ComparisonResult.lhs.id == Comparison.
                                    Item1.Criteria_ID
163                             && ComparisonResult.rhs.id == Comparison.
                                    Item2.Criteria_ID
164                             select ComparisonResult;
165             if (FoundResults.Count() > 0) Result = FoundResults.
                    FirstOrDefault();
166             else Result = null;
167         }
168         else Result = null;
169         if (Result != null)
170         {
171             Ratings.Add(new Rating()
172             {
173                 // LHS Criteria:
174                 LHS_Criteria = Comparison.Item1,
175                 LHS_Criteria_Value = (decimal)Result.lhs.value,
176                 // Rated by and when:
177                 Rated_By = SesInfo.Item3, /* Item3 = user */
178                 Rated_DtTm = DateTime.Now,
179                 // RHS Criteria:
180                 RHS_Criteria = Comparison.Item2,
181                 RHS_Criteria_Value = (decimal)Result.rhs.value,
182                 Slider_Pos = Result.lhs.sliderValue,
183                 Rating_ID = -1
184             });
```

```
185             }
186             else
187             {
188               Ratings.Add(new Rating()
189               {
190                 // LHS Criteria:
191                 LHS_Criteria = Comparison.Item1,
192                 LHS_Criteria_Value = 1m,
193                 // Rated by and when:
194                 Rated_By = SesInfo.Item3, /* Item3 = user */
195                 Rated_DtTm = DateTime.Now,
196                 // RHS Criteria:
197                 RHS_Criteria = Comparison.Item2,
198                 RHS_Criteria_Value = 1m,
199                 Slider_Pos = 500,
200                 Rating_ID = -1
201               });
202             }
203           }
204           // List is built, let's dump it to the DB:
205           SQL.Add_Update_Ratings(Ratings, SesInfo.Item1.Round_ID);
206         }
207         catch
208         {
209           Response.Redirect("Error.aspx");
210         }
211         // Let's go see the results:
212         if (SesInfo.Item1.Session_Is_Finished || SesInfo.Item3.
              IsAdministrator)
```

```
213             Response.Redirect("Results.aspx");
214         else
215             Response.Redirect("RoundEnd.aspx");
216       }
217     }
218 }
```

Listing 3: About.aspx

```
1 <%@ Page Title="About" Language="C#" MasterPageFile="~/Site.Master"
      AutoEventWireup="true" CodeBehind="About.aspx.cs" Inherits="Delphi.
      Web.About" %>
2
3 <asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="
      MainContent">
4     <hgroup class="title">
5         <h1><%: Title %>.</h1>
6         <h2>The PC Delphi System, version 1.0</h2>
7     </hgroup>
8
9     <article>
10        <p>
11            The PC Delphi System, a system for ranking alternatives
                  using Pairwise Comparisons and based on the workflow of
                  the Delphi System.
12        </p>
13        <p>
14            Questions or Comments? Please contact the author below.
15        </p>
16        <p>
```

```
17              Copyright 2014 Grant G. O. Duncan <a href="mailto:
                    gg_duncan@laurentian.ca">&lt;gg_duncan@laurentian.ca&gt
                    ;</a>
18          </p>
19      </article>
20
21      <aside>
22          <h3>PC Delphi</h3>
23          <ul>
24              <li><a runat="server" href="~/About.aspx">About</a></li>
25              <li><a runat="server" href="~/Contact.aspx">Contact</a></li
                    >
26          </ul>
27      </aside>
28  </asp:Content>
```

Listing 4: About.aspx.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.UI;
6  using System.Web.UI.WebControls;
7
8  namespace Delphi.Web
9  {
10   public partial class About : Page
11   {
12     protected void Page_Load(object sender, EventArgs e)
```

```
13      {
14
15      }
16    }
17  }
```

Listing 5: Consent.aspx

```
1  <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Consent.aspx.
       cs" Inherits="Delphi.Web.Consent" MasterPageFile="~/Site.Master" %>
2
3  <asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="
       MainContent">
4    <h1>Pleae carefully read the following before proceeding:</h1>
5    <ol class="round">
6      <li class="one">
7        <h2>Purpose:</h2>
8        <p>The purpose of this research is two-fold:
9          <li>
10           <ol>1. To assess the PC Delphi system in a real world
                 decision making process.</ol>
11           <ol>2. To improve a real world decision making process.</ol>
12         </li>
13       </p>
14     </li>
15     <li class="two">
16       <h2>Notes on Confidentialiaty and Anonymitiy:</h2>
17       <p>Please note that this application is confidential and
             anonymous. No identifying information will be collected, and
             your responses to the criteria ratings
```

```
18          will be kept anonymous.  At any point, you may request detailed
                information about how this data is being kept confidential,
                secure and anonymous by contacting
19          <a href="mailto:gg_duncan@laurentian.ca">Grant Duncan (
                gg_duncan@laurentian.ca)</a>.
20        </p>
21      </li>
22      <li class="three">
23        <h2>Your Right to non-participation and withdrawl:</h2>
24        <p>You have the right not to participate in this trial.  If you
              do wish to participate, please click the "I␣Consent␣to␣these␣
              terms␣and␣conditions" checkbox below and
25          click the button to proceed to the next screen.  Note that at
                any point, <i>you have the right to withdraw your consent to
                participate</i>, without consequence.
26          Furthermore, <i>you are under no obligation to use the rankings
                produced by the system for any decision making processes</i
                >.  The rankings produced are for
27          informational purposes only.
28        </p>
29      </li>
30      <li class="four">
31        <h2>You have a right to all results and write-ups:</h2>
32        <p>If you so desire, you have the right to all the results and to
              the any documents produced thereof.  If you would like to
              recieve the results of this study,
33          plese contact <a href="mailto:gg_duncan@laurentian.ca">Grant
                Duncan (gg_duncan@laurentian.ca)</a>.
34        </p>
```

```
35      </li>
36      <li class="five">
37        <h2>Alternate Contacts:</h2>
38        <p>Please note that you have the right to contact an official not
              attached to the research team regarding possible ethical
             issues or complaints about the reseach itself.
39          If you wish to make such inquiries or reports, please contact
              the following:<br />
40          <b>Research Ethics Officer, Laurentian University Research
              Office</b><br />
41          <b>705-675-1151, extensions 3213 or 2436.</b><br />
42          <b>Toll Free: 1-800-461-4030</b>
43          <b>email: <a href="mailto:ethics@laurentian.ca">
              Ethics@laurentian.ca</a></b>
44        </p>
45      </li>
46    </ol>
47    <asp:CheckBox ID="ChkConsent" runat="server" onchange="javascript:
        document.getElementById('MainContent_BtnContinue').disabled␣=␣
        false;" /> <b>I consent to these terms and conditions</b> &
        nbsp; <asp:Button ID="BtnContinue" runat="server" Text="Continue"
        OnClick="Consent_Click" Enabled="false"/>
48  </asp:Content>
```

Listing 6: Consent.aspx.cs

```
1  //
2  // Consent template code-behind.
3  // Ensures the user consented to the application otherwise redirects
       them
```

```
4  // to the appropriate area within the system.
5  // (c) 2014 <gg_duncan@laurentian.ca>
6  //
7  using System;
8  using System.Collections.Generic;
9  using System.Linq;
10 using System.Web;
11 using System.Web.UI;
12 using System.Web.UI.WebControls;
13 using Delphi.Web.Data;
14
15 namespace Delphi.Web
16 {
17   public partial class Consent : System.Web.UI.Page
18   {
19     private Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> _SesInfo = null;
20     private Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> SesInfo
21     {
22       get
23       {
24         if (_SesInfo == null)
25         {
26           _SesInfo = Session["SessionInfo"] as Tuple<Delphi.Data.
               Entities.Round, Delphi.Data.Entities.DelphiSession, Delphi
               .Data.Entities.User>;
27         }
28         return _SesInfo;
```

```
29          }
30          set
31          {
32            Session["SessionInfo"] = value;
33            _SesInfo = value;
34          }
35        }
36
37        private Delphi.Data.SQL.SqlClient _SQL = null;
38        private Delphi.Data.SQL.SqlClient SQL
39        {
40          get
41          {
42            if (_SQL == null)
43            {
44              _SQL = Session["SqlClient"] as Delphi.Data.SQL.SqlClient;
45            }
46            return _SQL;
47          }
48          set
49          {
50            Session["SqlClient"] = value;
51            _SQL = value;
52          }
53        }
54
55        protected void Page_Load(object sender, EventArgs e)
56        {
57          if (!Page.IsPostBack)
```

```
58          {
59              string UserGUID = Request.Params["GUID"];
60              SQL = new Delphi.Data.SQL.SqlClient();
61
62              SesInfo = SQL.Get_User_Rounds_by_GUID(UserGUID);
63
64              if (SesInfo.Item3.HasConsented)
65              {
66                  if (SesInfo.Item1.CanProceed) RedirectToRounds();
67                  else if (SesInfo.Item3.IsAdministrator || SesInfo.Item1.
                        Session_Is_Finished
68                      || SesInfo.Item1.Round_Number > 1 ) RedirectToResults();
69                  else RedirectToRoundEnd();
70              }
71          }
72      }
73
74      private void RedirectToRoundEnd()
75      {
76          Response.Redirect("RoundEnd.aspx");
77      }
78
79      private void RedirectToRounds()
80      {
81          Response.Redirect("Round.aspx");
82      }
83
84      private void RedirectToResults()
85      {
```

```
86          Response.Redirect("Results.aspx");
87      }
88
89      protected void Consent_Click(object sender, EventArgs e)
90      {
91          SQL.Set_Consent(SesInfo.Item3);
92          SesInfo.Item3.HasConsented = true;
93          SesInfo = SesInfo;
94          if (SesInfo.Item1.CanProceed) RedirectToRounds();
95          else if (SesInfo.Item3.IsAdministrator || SesInfo.Item1.
                Session_Is_Finished) RedirectToResults();
96          else RedirectToRoundEnd();
97      }
98  }
99 }
```

Listing 7: Contact.aspx

```
1 <%@ Page Title="Contact" Language="C#" MasterPageFile="~/Site.Master"
      AutoEventWireup="true" CodeBehind="Contact.aspx.cs" Inherits="Delphi
      .Web.Contact" %>
2
3 <asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="
      MainContent">
4      <hgroup class="title">
5          <h1><%: Title %>.</h1>
6          <h2>For questions or comments, please contact the author, Grant
               Duncan.</h2>
7      </hgroup>
8
```

```
 9      <section class="contact">
10          <header>
11              <h3>Email:</h3>
12          </header>
13          <p>
14              <span class="label">Grant Duncan, Msc. student in
                    Computational Sciences, Laurentian University:</span>
15              <span><a href="mailto:gg_duncan@laurentian.ca">
                    gg_duncan@laurentian.ca</a></span>
16          </p>
17      </section>
18  </asp:Content>
```

Listing 8: Contact.aspx.cs

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Web;
 5  using System.Web.UI;
 6  using System.Web.UI.WebControls;
 7
 8  namespace Delphi.Web
 9  {
10    public partial class Contact : Page
11    {
12      protected void Page_Load(object sender, EventArgs e)
13      {
14
15      }
```

```
16     }
17 }
```

Listing 9: Error.aspx

```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Error.aspx.cs
    " Inherits="Delphi.Web.Error" MasterPageFile="~/Site.Master" Title="
    An␣Error␣Has␣Occurred" %>
2 <asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="
    MainContent">
3     <hgroup class="title">
4         <h1><%: Title %>.</h1>
5         <h2>An Error Has Occurred!</h2>
6     </hgroup>
7
8     <section class="contact">
9         <header>
10            <h3>Error</h3>
11        </header>
12        <p>
13            <span class="label">An unexpected error has occurred.  If
                you require assistance or for further information please
                 contact:</span>
14            <span><a href="mailto:youremailhere">youremailhere</a></
                span>
15        </p>
16    </section>
17 </asp:Content>
```

Listing 10: Error.aspx.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.UI;
6  using System.Web.UI.WebControls;
7
8  namespace Delphi.Web
9  {
10   public partial class Error : System.Web.UI.Page
11   {
12     protected void Page_Load(object sender, EventArgs e)
13     {
14
15     }
16   }
17 }
```

Listing 11: Global.asax

```
1 <%@ Application Codebehind="Global.asax.cs" Inherits="Delphi.Web.Global
    " Language="C#" %>
```

Listing 12: Global.asax.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Routing;
```

```
6  using System.Web.Security;
7  using Delphi.Web;
8
9  namespace Delphi.Web
10 {
11   public class Global : HttpApplication
12   {
13     void Application_Start(object sender, EventArgs e)
14     {
15       // Code that runs on application startup
16       AuthConfig.RegisterOpenAuth();
17     }
18
19     void Application_End(object sender, EventArgs e)
20     {
21       //  Code that runs on application shutdown
22
23     }
24
25     void Application_Error(object sender, EventArgs e)
26     {
27       // Code that runs when an unhandled error occurs
28
29     }
30
31     protected void Session_Start(object sender, EventArgs e)
32     {
33       Session["null"] = "null";
34     }
```

```
35     }
36 }
```

Listing 13: Results.aspx

```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Results.aspx.
    cs" Inherits="Delphi.Web.Results" MasterPageFile="~/Site.Master" %>
2
3 <asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="
    MainContent">
4   <h1><% if (this.SesInfo.Item1.Session_Is_Finished) { %> Session is
      completed.  Displaying overall accepted rankings.
5     <% } else { %> Round #<%= this.SesInfo.Item1.Round_Number %>
        rankings are currently: <% } %></h1>
6   <h2>(<%= this.NumSubmitted %> results have been submitted.)</h2>
7   <ol class="round">
8     <asp:DataList ID="ResultsList" runat="server">
9       <ItemTemplate>
10        <li class="<%= this.LI_Classes[Iterator] %>">
11          <h2 <% if(OrderedCriteria[Iterator++].Item3) { %>style="color
              : red" <% } %> ><%#DataBinder.Eval(Container.DataItem, "
              Item1.Short_Description") %> (<%#DataBinder.Eval(Container
              .DataItem, "Item2") %>%) </h2>
12        </li>
13      </ItemTemplate>
14    </asp:DataList>
15  </ol>
16  <asp:HiddenField runat="server" ID="Hdn_II" />
17  <div class="floating-buttons-right">
18    <div id="g1"></div>
```

```
19      <asp:Label runat="server" ID="lbl_Warning" ForeColor="Red" Visible=
            "false">Warning: inconsistency is too high.</asp:Label><br />
20      <asp:Label runat="server" ID="lbl_Inconsis" Visible="false" /><br
            />
21      <asp:Button runat="server" ID="Btn_Finish" Text="Accept␣these␣
            Rankings" OnClick="Btn_Finish_Click" OnClientClick="return␣
            confirm('Accept␣these␣results␣and␣end␣this␣session?');"/>
22      <br />
23      <asp:Button runat="server" ID="Btn_NewRound" OnClick="
            Btn_NewRound_Click" />
24    </div>
25    <asp:PlaceHolder runat="server">
26      <script>
27        // Check if there's a gage on the page, if so
28        // set it up.
29        if ($("#g1").length != 0) {
30          IG = new JustGage({
31            id: "g1",
32            value: 0,
33            min: 0,
34            max: 100,
35            title: "Inconsistency",
36            label: "",
37            levelColorsGradient: false,
38            showMinMax: false
39          });
40          if ($("#MainContent_Hdn_II").length != 0) {
41            var ii = $("#MainContent_Hdn_II").val();
42            IG.refresh(Math.round(ii));
```

```
43            }
44          }
45        </script>
46    </asp:PlaceHolder>
47 </asp:Content>
```

Listing 14: Results.aspx.cs

```
1  //
2  // Results ASP template code behind:
3  // Handles displaying the results and firing the action to
4  // begin a new round, or end the session.
5  // (c) 2014
6  // <gg_duncan@laurentian.ca>
7  //
8  using System;
9  using System.Collections.Generic;
10 using System.Collections;
11 using System.Linq;
12 using System.Web;
13 using System.Web.UI;
14 using System.Web.UI.WebControls;
15 using Delphi.Data.Entities;
16 using PC;
17
18 namespace Delphi.Web
19 {
20   public partial class Results : System.Web.UI.Page
21   {
22     protected int NumSubmitted { get; private set; }
```

```
23
24      private Delphi.Data.SQL.SqlClient _SQL = null;
25      private Delphi.Data.SQL.SqlClient SQL
26      {
27        get
28        {
29          if (_SQL == null)
30          {
31            _SQL = Session["SqlClient"] as Delphi.Data.SQL.SqlClient;
32          }
33          return _SQL;
34        }
35        set
36        {
37          Session["SqlClient"] = value;
38          _SQL = value;
39        }
40      }
41
42      private Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
            DelphiSession, Delphi.Data.Entities.User> _SesInfo = null;
43      protected Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
            DelphiSession, Delphi.Data.Entities.User> SesInfo
44      {
45        get
46        {
47          if (_SesInfo == null)
48          {
```

```
49            _SesInfo = Session["SessionInfo"] as Tuple<Delphi.Data.
                  Entities.Round, Delphi.Data.Entities.DelphiSession, Delphi
                  .Data.Entities.User>;
50          }
51          return _SesInfo;
52        }
53        private set
54        {
55          Session["SessionInfo"] = value;
56          _SesInfo = value;
57        }
58      }
59
60      public List<Tuple<Criteria, decimal, bool>> OrderedCriteria { get;
            private set; }
61
62      // Used to make the results pretty:
63      private string[] _LI_Classes = new string[] { "one", "two", "three"
            , "four", "five", "six", "seven", "eight", "nine" };
64      public string[] LI_Classes { get { return _LI_Classes; } }
65      public int Iterator = 0;
66
67      protected void Page_Load(object sender, EventArgs e)
68      {
69        if (SesInfo == null) Response.Redirect("Error.aspx");
70        else
71        {
72          Btn_Finish.Visible = SesInfo.Item3.IsAdministrator && !SesInfo.
                Item1.Session_Is_Finished;
```

```
73          Btn_NewRound.Visible = SesInfo.Item3.IsAdministrator && !
                SesInfo.Item1.Session_Is_Finished;
74          Btn_NewRound.Text = "Begin␣Round␣" + (SesInfo.Item1.
                Round_Number + 1).ToString();
75          ResolveRatingData();
76          //ResolveRatingData_2();
77          ResultsList.DataSource = OrderedCriteria;
78          ResultsList.DataBind();
79        }
80      }
81      //
82      // Resolve Rating Data:
83      // 1. Roll up the ratings into a matrix of size [Comparisons, n]
84      // 2. Calculate the geometric means of this matrix to produce a
            List of size [comparisons]
85      // 3. Create a PC matrix of size nxn
86      // 4. Calculate the geometrix means of the PC matrix to produce the
            rankings
87      // 5. Normalize the rankings.
88      //
89      private void ResolveRatingData()
90      {
91        if (SesInfo != null)
92        {
93          if(!SesInfo.Item3.HasConsented) Response.Redirect(String.Format
                ("Consent.aspx?GUID={0}", SesInfo.Item3.GUID));
94          // Ratings:
95          List<Rating> ratings = SQL.GetRatingsByRoundID(SesInfo.Item1.
                Round_ID);
```

```
96          // criteria list:
97          List<Criteria> criteraList = SQL.GetCriteriaBySessionID(SesInfo
                .Item2.Session_ID);
98          // Helper vars:
99          decimal[] GeoMean = null;
100         decimal[,] PCMat = null, R = null;
101         decimal II = -1m;
102         int ii_i = 0, ii_j = 0, ii_k = 0;
103         // n  = number of criteria:
104         int n = Rating.NumCriteria;
105         // Array used to identify those criteria involved in the
                maximally inconsistent triad:
106         bool[] InconsistentTriads = new bool[n];
107         // number of comparisons given n:
108         int comps = (n * (n - 1)) / 2;
109         // accumulation indexes:
110         int Ridx1 = 0, Ridx2 = 0;
111         // Ensure no null ref errors:
112         if (ratings.Count > 0)
113         {
114           int LastUserID = ratings[0].Rated_By.User_ID;
115           R = new decimal[comps, ratings.Count / comps];
116           NumSubmitted = ratings.Count/comps;
117           // Accumulate the ratings:
118           for (int i = 0; i < ratings.Count; i++)
119           {
120             if (ratings[i].Rated_By.User_ID != LastUserID) { Ridx2++;
                    Ridx1 = 0; LastUserID = ratings[i].Rated_By.User_ID; }
121             if (ratings[i].Slider_Pos > 0)
```

```
122             R[Ridx1++, Ridx2] = ((1000m - ratings[i].Slider_Pos) / 10
                    m) / ((1.00m * ratings[i].Slider_Pos) / 10m);
123           else R[Ridx1++, Ridx2] = ratings[i].LHS_Criteria_Value /
                  ratings[i].RHS_Criteria_Value;
124         }
125         // Calculate the geomeans:
126         GeoMean = PCMath.Geometric_Means(R);
127         // Turn it into a PC matrix:
128         PCMat = PCMath.Build_PC_Mat(GeoMean, n);
129         // Calculate the new geomeans:
130         GeoMean = PCMath.Geometric_Means(PCMat);
131         // Calculate Koczkodaj's inconsistency indicator:
132         II = PCMath.Calculate_ii(PCMat, out ii_i, out ii_j, out ii_k)
              ;
133       }
134       // precentize it:
135       Hdn_II.Value = (II * 100.0m).ToString();
136       if(II > 0 && II > 1.0m/3.0m)
137       {
138         lbl_Warning.Visible = true;
139         lbl_Warning.Text += "␣(" + Math.Round(II * 100.0m).ToString()
              + ")";
140         for(int i=0; i<InconsistentTriads.Length; i++)
141         {
142           if (i == ii_i || i == ii_j || i == ii_k) InconsistentTriads
                [i] = true;
143           else InconsistentTriads[i] = false;
144         }
145       }
```

```
146         // Normalize the ranking's values:
147         decimal[] NormalizedMeans = PCMath.Normalize(GeoMean);
148         // Attach to their associated criteria (based on how we
                retrieve the info, we can safely assume that the
149         // ordering will remain the same, thus the use of the zip
                function):
150         OrderedCriteria = criteraList.Zip(NormalizedMeans, (criteria,
                geomeansvalue) => new Tuple<Criteria, decimal>(criteria,
                Math.Round(geomeansvalue, 0))).
151           Zip(InconsistentTriads, (criteria, icon)=> new Tuple<Criteria
                  , decimal, bool>(criteria.Item1, criteria.Item2, icon)).
                  ToList();
152         // Order them descending (highest ranking first):
153         OrderedCriteria = OrderedCriteria.OrderByDescending(o => o.
                Item2).ToList();
154       }
155       else Response.Redirect("Error.aspx");
156     }
157
158     protected void Btn_Finish_Click(object sender, EventArgs e)
159     {
160       SQL.End_Session(SesInfo.Item2, SesInfo.Item1);
161       Response.Redirect(String.Format("Consent.aspx?GUID={0}", SesInfo.
            Item3.GUID));
162     }
163
164     protected void Btn_NewRound_Click(object sender, EventArgs e)
165     {
166       SQL.New_Round(SesInfo.Item1);
```

```
167             Response.Redirect(String.Format("Consent.aspx?GUID={0}", SesInfo.
                    Item3.GUID));
168         }
169     }
170 }
```

Listing 15: Round.aspx

```
 1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Round.aspx.cs
       " Inherits="Delphi.Web.Round" MasterPageFile="~/Site.Master" %>
 2 <asp:Content runat="server" ID="FeaturedContent" ContentPlaceHolderID="
       FeaturedContent">
 3     <section class="featured">
 4         <div class="content-wrapper">
 5             <hgroup class="title">
 6                 <h1><%: Title %></h1>
 7                 <h2>Delphi Round</h2>
 8             </hgroup>
 9             <p style="margin-left:0px; overflow:auto; display:block;">
10                 Round #<%= this.RoundNumber %>. <%= this.SesInfo.Item2.
                       Description %>
11             </p>
12         </div>
13     </section>
14 </asp:Content>
15 <asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="
       MainContent">
16     <h3>Current Criteria to Assess</h3>
17   <input id="Hdn_CriteriaCount" type="hidden" value="<%= this.Crits.
         Length %>" />
```

```
18  <asp:HiddenField runat="server" ID="Hdn_II" />
19  <asp:HiddenField runat="server" ID="Comparison_Values" />
20      <%-- The CriteriaList are databound to table cells: --%>
21      <table style="width: 100%">
22        <tbody>
23          <asp:DataList ID="RoundItems" runat="server">
24            <ItemTemplate>
25              <tr>
26                <%-- Lhs: --%>
27                <td style="font-size: large">
28                  <label style="font-size: 100%" id="slider<%# this.
                        IDIterator.ToString() %>_label1"><%#DataBinder.
                        Eval(Container.DataItem, "Item1.Short_Description"
                        )%></label>
29                  <input id="slider<%#(this.IDIterator).ToString() %>
                        _iterator" type="hidden" value="<%#(this.
                        IDIterator).ToString() %>"/>
30                  <input id="slider<%#(this.IDIterator).ToString() %>
                        _LHS_Criteria_ID" type="hidden" value="<%#
                        DataBinder.Eval(Container.DataItem, "Item1.
                        Criteria_ID") %>"/>
31                  <input id="slider<%#(this.IDIterator).ToString() %>
                        _description1" type="hidden" value="<%#DataBinder.
                        Eval(Container.DataItem, "Item1.Description") %>"
                        />
32                  <input id="slider<%#(this.IDIterator).ToString() %>
                        _values" type="hidden" value="<%#DataBinder.Eval(
                        Container.DataItem, "Item3") %>" />
```

```
33                  <input id="slider<%#(this.IDIterator).ToString()␣%>
                        _Rangemin" type="hidden" value="<%#DataBinder.Eval
                        (Container.DataItem,␣"Item1.CriteriaRange.
                        Low_Range")␣%>" />
34                  <input id="slider<%#(this.IDIterator).ToString()␣%>
                        _Rangemax" type="hidden" value="<%#DataBinder.Eval
                        (Container.DataItem,␣"Item1.CriteriaRange.
                        High_Range")␣%>" />
35                  <input id="slider<%#(this.IDIterator).ToString()␣%>
                        _Rangestep" type="hidden" value="<%#DataBinder.
                        Eval(Container.DataItem,␣"Item1.CriteriaRange.Step
                        ")␣%>" />
36                </td>
37                <td>
38                  <%-- Slider: --%>
39                  <div id="slider<%#␣this.IDIterator.ToString()␣%>"
                        class="slider" />
40                </td>
41                <%-- Rhs: --%>
42                <td style="font-size:␣large">
43                  <input id="slider<%#(this.IDIterator).ToString()␣%>
                        _RHS_Criteria_ID" type="hidden" value="<%#
                        DataBinder.Eval(Container.DataItem,␣"Item2.
                        Criteria_ID")␣%>"/>
44                  <label style="font-size:␣100%" id="slider<%#␣(this.
                        IDIterator).ToString()␣%>_label2"><%#DataBinder.
                        Eval(Container.DataItem, "Item2.Short_Description"
                        )%></label>
```

```
45                      <input id="slider<%#(this.IDIterator++).ToString()␣%>
                            _description2" type="hidden" value="<%#DataBinder.
                            Eval(Container.DataItem,␣"Item2.Description")␣%>"
                            />
46                    </td>
47                  </tr>
48                </ItemTemplate>
49              </asp:DataList>
50            </tbody>
51            <tfoot>
52              <tr>
53                <td>
54                  <textarea id="txt_area"></textarea>
55                </td>
56              </tr>
57            </tfoot>
58          </table>
59    <asp:Button runat="server" ID="Btn_Submit" OnClick="Btn_Submit_Click"
          Text="Submit␣Results" />
60  </asp:Content>
```

Listing 16: Round.aspx.cs

```
1  //
2  // Rounds.aspx.cs
3  // (c) 2014 Grant G. O. Duncan <gg_duncan@laurentian.ca>
4  // Code behind file for the Round.aspx webform.
5  // This page runs a round for the given session.
6  //
7  using System;
```

```
8  using System.Collections.Generic;
9  using System.Linq;
10 using System.Web;
11 using System.Web.UI;
12 using System.Web.UI.WebControls;
13 using Delphi.Data.Entities;
14
15 namespace Delphi.Web
16 {
17   public partial class Round : System.Web.UI.Page
18   {
19     private Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> _SesInfo = null;
20     protected Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> SesInfo
21     {
22       get
23       {
24         if (_SesInfo == null)
25         {
26           _SesInfo = Session["SessionInfo"] as Tuple<Delphi.Data.
                Entities.Round, Delphi.Data.Entities.DelphiSession, Delphi
                .Data.Entities.User>;
27         }
28         return _SesInfo;
29       }
30       set
31       {
32         Session["SessionInfo"] = value;
```

```
33          _SesInfo = value;
34        }
35      }
36
37      private Delphi.Data.SQL.SqlClient _SQL = null;
38      private Delphi.Data.SQL.SqlClient SQL
39      {
40        get
41        {
42          if (_SQL == null)
43          {
44            _SQL = Session["SqlClient"] as Delphi.Data.SQL.SqlClient;
45          }
46          return _SQL;
47        }
48        set
49        {
50          Session["SqlClient"] = value;
51          _SQL = value;
52        }
53      }
54
55      protected int RoundNumber { get { return SesInfo.Item1.Round_Number
            ; } }
56      private List<Tuple<Criteria, Criteria>> _CriteriaList = null;
57      protected List<Tuple<Criteria, Criteria>> CriteriaList
58      {
59        get
60        {
```

```
61          if (_CriteriaList == null)
62          {
63            _CriteriaList = Session["CriteriaList"] as List<Tuple<
                  Criteria, Criteria>>;
64          }
65          return _CriteriaList;
66        }
67        private set
68        {
69          Session["CriteriaList"] = value;
70          _CriteriaList = value;
71        }
72      }
73
74      private List<Tuple<Criteria, Criteria, string>>
           _CriteriaListWithPosition = null;
75      protected List<Tuple<Criteria, Criteria,string>>
           CriteriaListWithPosition
76      {
77        get
78        {
79          if (_CriteriaListWithPosition == null)
80          {
81            _CriteriaListWithPosition = Session["CriteriaListWithPosition
                  "] as List<Tuple<Criteria, Criteria,string>>;
82          }
83          return _CriteriaListWithPosition;
84        }
85        private set
```

```
86          {
87              Session["CriteriaListWithPosition"] = value;
88              _CriteriaListWithPosition = value;
89          }
90      }
91
92      protected int IDIterator { get; set; }
93      private Delphi.Data.SQL.SqlClient SqlClient = new Delphi.Data.SQL.
            SqlClient();
94      protected Criteria[] Crits { get; private set; }
95
96      protected void Page_Load(object sender, EventArgs e)
97      {
98          if (!this.IsPostBack)
99          {
100             IDIterator = 0;
101             // Has some session info, so probably can proceed:
102             if (SesInfo != null && SesInfo.Item3.HasConsented)
103             {
104                 Crits = SqlClient.GetCriteriaBySessionID(SesInfo.Item2.
                        Session_ID).ToArray();
105                 CriteriaList = Criteria.Make_PC_Criteria(Crits);
106                 Get_Last_Round_Position_Data();
107                 RoundItems.DataSource = CriteriaListWithPosition;
108                 RoundItems.DataBind();
109             }
110             else
111             {
112                 Response.Redirect("Consent.aspx");
```

```
113             }
114         }
115     }
116
117     private void Get_Last_Round_Position_Data()
118     {
119         CriteriaListWithPosition = new List<Tuple<Criteria, Criteria,
                string>>();
120         List<Tuple<int, int, int>> PrevRatings = SQL.
                GetLastRoundBySessionID(SesInfo.Item2.Session_ID);
121         IEnumerable<Tuple<int, int, int>> AvgPosition = null;
122         foreach (Tuple<Criteria, Criteria> crits in CriteriaList)
123         {
124             if(PrevRatings.Count > 0)
125             {
126                 AvgPosition = from Tuple<int, int, int> t in PrevRatings
127                               where crits.Item1.Criteria_ID == t.Item1 &&
128                                     crits.Item2.Criteria_ID == t.Item2
129                               select t;
130                 if (AvgPosition.Count() > 0)
131                     CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                            string>(crits.Item1, crits.Item2, "[500," + AvgPosition.
                            First().Item3.ToString() + "]"));
132                 else
133                     CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                            string>(crits.Item1, crits.Item2, "[500]"));
134             }
135             else
```

```
136             CriteriaListWithPosition.Add(new Tuple<Criteria, Criteria,
                    string>(crits.Item1, crits.Item2, "[500]"));
137         }
138     }
139
140     protected void Btn_Submit_Click(object sender, EventArgs e)
141     {
142         try
143         {
144             IEnumerable<Delphi.Web.Data.ComparisonData> FoundResults = null
                    ;
145             Delphi.Web.Data.ComparisonData Result = null;
146             List<Rating> Ratings = new List<Rating>();
147
148             // Deserialize the JSON comparison data:
149             Delphi.Web.Data.ComparisonData[] ComparisonResults = Delphi.Web
                    .Data.ComparisonData.Deserialize(Comparison_Values.Value);
150             //
151             // Grab the results:
152             // Constructs a list of ratings
153             // Resolves for all criteria:criteria comparisons, using 1 for
                    those items that
154             // do not exist in the submitted result set
155             //
156             foreach (Tuple<Criteria, Criteria> Comparison in CriteriaList)
157             {
158                 if (ComparisonResults != null)
159                 {
```

```
160             FoundResults = from Delphi.Web.Data.ComparisonData
                    ComparisonResult in ComparisonResults
161                             where ComparisonResult != null
162                             && ComparisonResult.lhs.id == Comparison.
                                    Item1.Criteria_ID
163                             && ComparisonResult.rhs.id == Comparison.
                                    Item2.Criteria_ID
164                             select ComparisonResult;
165             if (FoundResults.Count() > 0) Result = FoundResults.
                    FirstOrDefault();
166             else Result = null;
167         }
168         else Result = null;
169         if (Result != null)
170         {
171             Ratings.Add(new Rating()
172             {
173                 // LHS Criteria:
174                 LHS_Criteria = Comparison.Item1,
175                 LHS_Criteria_Value = (decimal)Result.lhs.value,
176                 // Rated by and when:
177                 Rated_By = SesInfo.Item3, /* Item3 = user */
178                 Rated_DtTm = DateTime.Now,
179                 // RHS Criteria:
180                 RHS_Criteria = Comparison.Item2,
181                 RHS_Criteria_Value = (decimal)Result.rhs.value,
182                 Slider_Pos = Result.lhs.sliderValue,
183                 Rating_ID = -1
184             });
```

```
185             }
186             else
187             {
188               Ratings.Add(new Rating()
189               {
190                 // LHS Criteria:
191                 LHS_Criteria = Comparison.Item1,
192                 LHS_Criteria_Value = 1m,
193                 // Rated by and when:
194                 Rated_By = SesInfo.Item3, /* Item3 = user */
195                 Rated_DtTm = DateTime.Now,
196                 // RHS Criteria:
197                 RHS_Criteria = Comparison.Item2,
198                 RHS_Criteria_Value = 1m,
199                 Slider_Pos = 500,
200                 Rating_ID = -1
201               });
202             }
203           }
204           // List is built, let's dump it to the DB:
205           SQL.Add_Update_Ratings(Ratings, SesInfo.Item1.Round_ID);
206         }
207         catch
208         {
209           Response.Redirect("Error.aspx");
210         }
211         // Let's go see the results:
212         if (SesInfo.Item1.Session_Is_Finished || SesInfo.Item3.
              IsAdministrator)
```

```
213             Response.Redirect("Results.aspx");
214         else
215             Response.Redirect("RoundEnd.aspx");
216       }
217     }
218 }
```

Listing 17: RoundEnd.aspx

```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="RoundEnd.aspx
      .cs" Inherits="Delphi.Web.RoundEnd" MasterPageFile="~/Site.Master"
      Title="Submission␣Accepted" %>
2
3 <asp:Content runat="server" ID="FeaturedContent" ContentPlaceHolderID="
      FeaturedContent">
4   <section class="featured">
5     <div class="content-wrapper">
6       <hgroup class="title">
7         <h1>Thank you for your Submission!</h1>
8       </hgroup>
9     </div>
10  </section>
11 </asp:Content>
12 <asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="
      MainContent">
13   <p style="margin-left:␣0px;␣overflow:␣auto;␣display:␣block;">
14     Your submission of ratings for round #<%= this.SesInfo.Item1.
         Round_Number.ToString() %> has been accepted.
15   </p>
16 </asp:Content>
```

Listing 18: RoundEnd.aspx.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.UI;
6  using System.Web.UI.WebControls;
7
8  namespace Delphi.Web
9  {
10   public partial class RoundEnd : System.Web.UI.Page
11   {
12     private Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> _SesInfo = null;
13     protected Tuple<Delphi.Data.Entities.Round, Delphi.Data.Entities.
         DelphiSession, Delphi.Data.Entities.User> SesInfo
14     {
15       get
16       {
17         if (_SesInfo == null)
18         {
19           _SesInfo = Session["SessionInfo"] as Tuple<Delphi.Data.
               Entities.Round, Delphi.Data.Entities.DelphiSession, Delphi
               .Data.Entities.User>;
20         }
21         return _SesInfo;
22       }
23       private set
24       {
```

```
25            Session["SessionInfo"] = value;
26            _SesInfo = value;
27          }
28        }
29
30        protected void Page_Load(object sender, EventArgs e)
31        {
32          if (SesInfo == null || !SesInfo.Item3.HasConsented)
33          {
34            Response.Redirect("Consent.aspx");
35          }
36        }
37      }
38  }
```

Listing 19: Site.Master

```
1  <%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master
       .cs" Inherits="Delphi.Web.SiteMaster" %>
2
3  <!DOCTYPE html>
4  <html lang="en">
5  <head runat="server">
6    <meta charset="utf-8" />
7    <title><%: Page.Title %></title>
8    <link href="~/Content/Site.css" rel="stylesheet" />
9    <link href="~/Content/PCDelphi.css" rel="stylesheet" />
10   <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/
         smoothness/jquery-ui.css" />
11   <link href="favicon.ico" rel="shortcut␣icon" type="image/x-icon" />
```

```
12    <link rel="stylesheet" href="~/Content/Slider.css" />
13    <asp:PlaceHolder runat="server">
14      <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
15      <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.min.js"></
            script>
16      <script src="Scripts/jquery.ui.touch-punch.min.js"></script>
17      <script src="<%: ResolveUrl("~/Scripts/modernizr-2.5.3.js") %>"></
            script>
18      <script src="Scripts/justgage.1.0.1.min.js"></script>
19      <script src="Scripts/raphael.2.1.0.min.js"></script>
20      <script src="Scripts/Sliders.js?10"></script>
21      <script src="Scripts/PC.js"></script>
22      <!-- Dom Ready Script:                                          -->
23      <!-- Set up any first pass scripts and initializer here -->
24      <!--                                                           -->
25      <script>
26        var IG;
27        var Comparisons = [];
28        var ComparisonsIndex = 0;
29        $(function () {
30          // Initialize and create the Sliders:
31          Init_Sliders();
32        });
33      </script>
34    </asp:PlaceHolder>
35    <meta name="viewport" content="width=device-width" />
36    <asp:ContentPlaceHolder runat="server" ID="HeadContent" />
37  </head>
38  <body>
```

```
39  <form runat="server">
40    <asp:ScriptManager runat="server">
41      <Scripts>
42        <asp:ScriptReference Name="jquery" />
43        <asp:ScriptReference Name="jquery.ui.combined" />
44      </Scripts>
45    </asp:ScriptManager>
46    <header>
47      <div class="content-wrapper">
48        <div class="float-left">
49          <p class="site-title">PC-Delphi</p>
50        </div>
51        <div class="float-right">
52          <nav>
53            <ul id="menu">
54              <li><a runat="server" href="~/About.aspx">About</a></li>
55              <li><a runat="server" href="~/Contact.aspx">Contact</a></
                    li>
56            </ul>
57          </nav>
58        </div>
59      </div>
60    </header>
61    <div id="body">
62      <asp:ContentPlaceHolder runat="server" ID="FeaturedContent" />
63      <section class="content-wrapper␣main-content␣clear-fix">
64        <asp:ContentPlaceHolder runat="server" ID="MainContent" />
65      </section>
66    </div>
```

```
67      <footer>
68        <div class="content-wrapper">
69          <div class="float-left">
70            <p>&copy; <%: DateTime.Now.Year %> - Grant G. O. Duncan</p>
71          </div>
72        </div>
73      </footer>
74    </form>
75  </body>
76  </html>
```

Listing 20: Site.Master.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Security;
6  using System.Web.UI;
7  using System.Web.UI.WebControls;
8
9  namespace Delphi.Web
10 {
11   public partial class SiteMaster : MasterPage
12   {
13     private const string AntiXsrfTokenKey = "__AntiXsrfToken";
14     private const string AntiXsrfUserNameKey = "__AntiXsrfUserName";
15     private string _antiXsrfTokenValue;
16
17     protected void Page_Init(object sender, EventArgs e)
```

```
18      {
19          // The code below helps to protect against XSRF attacks
20          var requestCookie = Request.Cookies[AntiXsrfTokenKey];
21          Guid requestCookieGuidValue;
22          if (requestCookie != null && Guid.TryParse(requestCookie.Value,
                out requestCookieGuidValue))
23          {
24              // Use the Anti-XSRF token from the cookie
25              _antiXsrfTokenValue = requestCookie.Value;
26              Page.ViewStateUserKey = _antiXsrfTokenValue;
27          }
28          else
29          {
30              // Generate a new Anti-XSRF token and save to the cookie
31              _antiXsrfTokenValue = Guid.NewGuid().ToString("N");
32              Page.ViewStateUserKey = _antiXsrfTokenValue;
33
34              var responseCookie = new HttpCookie(AntiXsrfTokenKey)
35              {
36                  HttpOnly = true,
37                  Value = _antiXsrfTokenValue
38              };
39              if (FormsAuthentication.RequireSSL && Request.
                    IsSecureConnection)
40              {
41                  responseCookie.Secure = true;
42              }
43              Response.Cookies.Set(responseCookie);
44          }
```

```
45
46          Page.PreLoad += master_Page_PreLoad;
47      }
48
49      protected void master_Page_PreLoad(object sender, EventArgs e)
50      {
51          if (!IsPostBack)
52          {
53              // Set Anti-XSRF token
54              ViewState[AntiXsrfTokenKey] = Page.ViewStateUserKey;
55              ViewState[AntiXsrfUserNameKey] = Context.User.Identity.Name ??
                    String.Empty;
56          }
57          else
58          {
59              // Validate the Anti-XSRF token
60              if ((string)ViewState[AntiXsrfTokenKey] != _antiXsrfTokenValue
61                  || (string)ViewState[AntiXsrfUserNameKey] != (Context.User.
                        Identity.Name ?? String.Empty))
62              {
63                  throw new InvalidOperationException("Validation␣of␣Anti-XSRF␣
                        token␣failed.");
64              }
65          }
66      }
67
68      protected void Page_Load(object sender, EventArgs e)
69      {
70
```

```
71        }
72    }
73 }
```

Listing 21: Web.config

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <configuration>
3    <configSections>
4      <section name="SQLSettings" type="Delphi.Data.SQL.SQLSettings,␣
           Delphi.Data"/>
5    </configSections>
6    <!-- SQL Settings Custom configuration item: -->
7    <!-- See Delphi.Data.SQL.SQLSettings.cs for more information -->
8    <SQLSettings ConnectionString="Data␣Source=.;Initial␣Catalog=
       Delphi_PC;User␣ID=app_Delphi_PC_User;Password=
       e2b491a53e694fa3b65d5dc43b67d220"/>
9    <system.web>
10     <compilation debug="true" targetFramework="4.0" />
11     <authentication mode="Forms">
12       <!-- Main authentication page: -->
13       <forms loginUrl="~/Consent.aspx" timeout="2880" />
14     </authentication>
15     <sessionState mode="InProc" customProvider="DefaultSessionProvider"
         >
16       <providers>
17         <add name="DefaultSessionProvider" type="System.Web.Providers.
             DefaultSessionStateProvider,␣System.Web.Providers,␣Version
             =1.0.0.0,␣Culture=neutral,␣PublicKeyToken=31bf3856ad364e35"
             connectionStringName="DefaultConnection" />
```

```
18          </providers>
19        </sessionState>
20    </system.web>
21    <system.webServer>
22        <modules runAllManagedModulesForAllRequests="true" />
23    </system.webServer>
24 </configuration>
```

Listing 22: PCDelphi.css

```
1 /* PC Delphic specific styles: */
2 .floating-buttons-right {
3     position:absolute;
4     top: 150px;
5     right: 200px;
6     float: right;
7 }
```

Listing 23: Slider.css

```
1 /*
2  Slider Styles:
3 */
4 .slider {
5     height: 20px;
6     width: 500px;
7     background-image: none;
8 }
9
10 .ui-slider .ui-widget-header {
11     background-image: none;
```

```
12    background-color: white;
13    border: none;
14  }
15
16  .ui-slider .ui-widget {
17    background-image: none;
18    border: none;
19    z-index: 5
20  }
21
22  .ui-slider .ui-slider-handle {
23    height: 25px;
24    width: 1px;
25    background-image: none;
26    background-color: grey;
27    border: 1px solid;
28    border-color: black;
29    z-index: 25
30  }
31
32  .slider .stay {
33    height: 25px;
34    width: 1px;
35    background-image: none;
36    background-color: white;
37    border: 1px solid;
38    border-color: red;
39    z-index: 10;
40  }
```

Listing 24: ComparisonData.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using Newtonsoft.Json;
6
7  namespace Delphi.Web.Data
8  {
9    public class Comparison
10   {
11     public int id { get; set; }
12     public double value { get; set; }
13     public int sliderValue { get; set; }
14   }
15
16   public class ComparisonData
17   {
18     public Comparison lhs { get; set; }
19     public Comparison rhs { get; set; }
20
21     public static ComparisonData[] Deserialize(string Data)
22     {
23       ComparisonData[] retval = null;
24       retval = JsonConvert.DeserializeObject<ComparisonData[]>(Data);
25       return retval;
26     }
27   }
28 }
```

Listing 25: SessionInfo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Delphi.Data.Entities;

namespace Delphi.Web.Data
{
    public class SessionInfo
    {
        public int Session_ID { get; set; }
        public int Round_Number { get; set; }
        public User CurrentUser { get; set; }
    }
}
```

Listing 26: PC.js

```

//
// CreatePC(B, n)
// Creates a PC matrix made up of comparions B
// derived from n criteria
//
function CreatePC(B, n) {
    var k = 0;
    var A = new Array(n);
    for (i = 0; i < n; i++) {
        A[i] = new Array(n);
```

```
12    }
13
14    for (i = 0; i < n; i++) {
15      for (j = 0; j < n; j++) {
16        if (i == j) A[i][j] = 1;
17        else if (i < j) A[i][j] = B[k++];
18        else if (i > j) A[i][j] = 1.0 / A[j][i];
19      }
20    }
21    //alert(A);
22    return A;
23  }
24
25  //
26  //II(els);
27  //
28  // Produces the expected results, but doesn't seem right...
29  //
30  function II(A) {
31    var ii = 0;
32    var ii_t = 0;
33    var k = 0;
34    var n = A[0].length;
35
36    for (j = n - 1; j >= 0; j--) {
37      k = j - 1;
38      for (i = j - 2; i >= 0; i--) {
39        ii_t = Math.min(
40            Math.abs((1.0 - A[i][j]) / (A[i][j] * A[k][j] * A[k][j])),
```

```
41              Math.abs(1.0 - A[i][k] * A[k][j] / A[i][j])
42          );
43          if (ii_t > ii) ii = ii_t;
44        }
45        k--;
46      }
47      return ii;
48  }
```

Listing 27: Sliders.js

```
1  //
2  // Sliders.js
3  // (C) 2014 Grant G. O. Duncan <gg_duncan@laurentian.ca>
4  // This script file handles slider interaction on the page.
5  //
6
7  //
8  // Init_Sliders()
9  // Initializes and creates the sliders, as well as handling the
10 // slider's "slide" event.
11 // Note: meant to be called from the page's DOM ready script
12 //
13 function Init_Sliders() {
14   // Find all the sliders:
15   var sliders = $(".slider");
16   // Initialize them:
17   sliders.each(function () {
18     $(this).empty().slider({
19       orientation: "horizontal",
```

```
20          range: "min",
21          min: $("#" + this.id + "_Rangemin").val() == undefined ? 2 :
                parseInt($("#" + this.id + "_Rangemin").val()),
22          max: $("#" + this.id + "_Rangemax").val() == undefined ? 999 :
                parseInt($("#" + this.id + "_Rangemax").val()),
23          values: eval($("#" + this.id + "_values").val()),
24          step: $("#" + this.id + "_Rangestep").val() == undefined ? 1 :
                parseInt($("#" + this.id + "_Rangestep").val()),
25          //
26          // Start function:
27          // Used to ensure that the second handle does not move.
28          //
29          start: function (event, ui) {
30            if ($(ui.handle).hasClass("stay"))
31              return false;
32          },
33          //
34          // onslide function:
35          // Get the slider's ID, find the two attached labels,
36          // apply the sizing based on the slider's departure from the
37          // middle.
38          // Uses a logarithmic scale for resizing the labels:
39          // A/B = log(1000-sliderValue)/log(sliderValue)
40          //
41          // Also creates a JSON structure that will return the values to
                the server
42          // on postback.
43          //
44          slide: function (event, ui) {
```

```
45          var sliders2 = $(".slider");
46          var CriteriaCount = $("#Hdn_CriteriaCount").val();
47          var Judgments = new Array(sliders2.length);
48          var Value = 1;
49          // Build the PC matrix:
50          for (var i = 0; i < sliders2.length; i++) {
51            if (sliders2[i].id == this.id)
52              Judgments[i] = Math.log(1000 - ui.value) / Math.log(ui.
                    value);
53            else {
54              Value = $("#" + sliders2[i].id).slider("value");
55              Judgments[i] = Math.log(1000 - Value) / Math.log(Value);
56            }
57          }
58          // Set the values for this instance:
59          var Iterator = parseInt($("#" + this.id + "_iterator").val());
60          var LHS_Criteria_ID = parseInt($("#" + this.id + "
                _LHS_Criteria_ID").val());
61          var RHS_Criteria_ID = parseInt($("#" + this.id + "
                _RHS_Criteria_ID").val());
62          var LHS = { id: LHS_Criteria_ID, value: Math.log(1000 - ui.
                value), sliderValue: ui.value };
63          var RHS = { id: RHS_Criteria_ID, value: Math.log(ui.value),
                sliderValue: ui.value };
64          Comparisons[Iterator] = { lhs: LHS, rhs: RHS };
65          $("#MainContent_Comparison_Values").val(JSON.stringify(
                Comparisons));
66          // Set the labels
67          var A2B = Math.log(1000 - ui.value) / Math.log(ui.value);
```

```
68            var B2A = 1 / A2B;
69            var label1 = $("#" + this.id + "_label1");
70            var label2 = $("#" + this.id + "_label2");
71            label1[0].style.fontSize = (A2B * 100.0).toString() + "%";
72            label2[0].style.fontSize = (B2A * 100.0).toString() + "%";
73            $("#txt_area").val("Compare " + $("#" + this.id + "
                  _description1").val() + " against " + $("#" + this.id + "
                  _description2").val());
74          }
75        });
76        //
77        // Set up the handle for any multi sliders that show the average
              slider position
78        // From a previous round:
79        if (($("#" + this.id + "_values").val()) != "[500]") {
80          $("#" + this.id + " .ui-slider-handle:last").addClass("stay");
81        }
82      });
83  }
```

A.2 Delphi.Data Source Code Listing

The Delphi.Data library comprises the source code that handles the data access layer (or DAL) of the PC Delphi reference implementation. This library is designed to be compiled into a DLL for inclusion into the Delphi.Web project, though in actuality can serve as a DAL for any .NET application wishing to communicate with the PC Delphi SQL database layer. The source is divided into several logical namespaces: the Entities namespace contains the classes that map to various SQL table entities as well as providing logic for their manipulation. The Exceptions namespace contains exception class wrappers so as to enable more accurate error handling. The SQL namespace

contains the SQLClient code that enables communication with the back-end SQL database.

A.2.1 Entities Namespace

The following 7 files make up the various entities in use by the reference implementation.

Listing 28: Users Entity

```
1  //
2  // User entity.
3  // (c) 2014 <gg_duncan@laurentian.ca>
4  //
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9
10 namespace Delphi.Data.Entities
11 {
12   public class User
13   {
14     public int User_ID { get; set; }
15     public string GUID { get; set; }
16     public bool HasConsented { get; set; }
17     public DateTime Consent_DtTm { get; set; }
18     public bool IsAdministrator { get; set; }
19   }
20 }
```

Listing 29: Rounds Entity

```
1  //
```

```
2  // Entity that represents a round instance.
3  // If the round is finished, CanProceed with be false and Ratings will
       be filled
4  // with the round's ratings.
5  // (c) 2014 <gg_duncan@laurentian.ca>
6  //
7  using System;
8  using System.Collections.Generic;
9  using System.Linq;
10 using System.Text;
11
12 namespace Delphi.Data.Entities
13 {
14   public class Round
15   {
16     public int Round_ID { get; set; }
17     public DateTime Start_DtTm { get; set; }
18     public DateTime End_DtTm { get; set; }
19     public int Round_Number { get; set; }
20     public List<Rating> Ratings { get; set; }
21     public bool CanProceed { get; set; }
22     public bool Session_Is_Finished { get; set; }
23
24     public Round() { Ratings = new List<Rating>(); Round_ID = -1; }
25   }
26 }
```

Listing 30: Ratings Entity

```
1  //
```

```
2  // An individual rating: the rated values between the LHS criteria
3  // and the RHS criteria.
4  // (c) 2014 <gg_duncan@laurentian.ca>
5  //
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10
11 namespace Delphi.Data.Entities
12 {
13   public class Rating
14   {
15     public int Rating_ID { get; set; }
16     public Criteria LHS_Criteria { get; set; }
17     public decimal LHS_Criteria_Value { get; set; }
18     public Criteria RHS_Criteria { get; set; }
19     public decimal RHS_Criteria_Value { get; set; }
20     public User Rated_By { get; set; }
21     public DateTime Rated_DtTm { get; set; }
22     public static int NumCriteria { get; set; }
23     public int Slider_Pos { get; set; }
24   }
25 }
```

Listing 31: Range Entity

```
1  //
2  // The range of the comparisons:
3  // (c) 2014 <gg_duncan@laurentian.ca>
```

```
4  //
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9
10 namespace Delphi.Data.Entities
11 {
12   public class Range
13   {
14     public int Range_ID { get; set; }
15     public double High_Range { get; set; }
16     public double Low_Range { get; set; }
17     public double Step { get; set; }
18   }
19 }
```

Listing 32: Criteria Entity

```
1  //
2  // Criteria entity:
3  // Represents a criteria retrieved from the DB:
4  // (c) 2014 <gg_duncan@laurentian.ca>
5  //
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using Delphi.Data;
11
```

```
12 namespace Delphi.Data.Entities
13 {
14   public partial class Criteria
15   {
16     public int Criteria_ID { get; set; }
17     public string Description { get;set; }
18     public string Short_Description { get; set; }
19     public DateTime Create_DtTm { get; set; }
20     public User CreatedBy { get; set; }
21     public Range CriteriaRange { get;set; }
22
23     //
24     // Creates a list of Criteria tuples {LHS, RHS} from an array of
           criteria.
25     // The resulting tuples represent the PC comparisons between each
           criteria (one to another).
26     // Note that we makes assumption of reciprocity
27     //
28     public static List<Tuple<Criteria, Criteria>> Make_PC_Criteria(
           params Criteria[] Crits)
29     {
30       List<Tuple<Criteria, Criteria>> retval = new List<Tuple<Criteria,
             Criteria>>();
31
32       for (int i = 0; i < Crits.Length; i++)
33       {
34         for (int j = i+1; j < Crits.Length; j++)
35         {
```

```
36              retval.Add(new Tuple<Criteria, Criteria>(Crits[i], Crits[j]))
                    ;
37            }
38          }
39
40          return retval;
41        }
42    }
43 }
```

Listing 33: Criteria UI Extension

```
1  //
2  // UI Specific properties that help the display and usage of the
       criteria:
3  // (c) 2014 <gg_duncan@laurentian.ca>
4  //
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9
10 namespace Delphi.Data.Entities
11 {
12   public partial class Criteria
13   {
14     public string UI_AverageSliderPositionList { get; set; }
15   }
16 }
```

Listing 34: Delphi Session Entity

```
1  //
2  // DelphiSession.cs
3  // Represents a session in the pcdelphi process
4  // (c) 2014 <gg_duncan@laurentian.ca>
5  //
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10
11 namespace Delphi.Data.Entities
12 {
13   public class DelphiSession
14   {
15     public int Session_ID { get; set; }
16     public DateTime Create_DtTm { get; set; }
17     public int Round_Threshold { get; set; }
18     public User CreatedBy { get; set; }
19     public string Description { get; set; }
20     public string Short_Description { get; set; }
21     public DateTime Started_DtTm { get; set; }
22     public DateTime Ended_DtTm { get; set; }
23     public List<Round> Rounds { get; set; }
24
25     public DelphiSession() { Rounds = new List<Round>(); }
26   }
27 }
```

A.2.2 Exceptions Namespace

A single file wraps exceptions from the SQL namespace in order to implement better exception handling and identification from calling modules.

Listing 35: Delphi SQL Client Exception wrapper

```
1  //
2  // Delphi SQL exception wrapper:
3  // (c) 2014 <gg_duncan@laurentian.ca>
4  //
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9
10 namespace Delphi.Data.Exceptions
11 {
12   public class DelphiSqlClientException : Exception
13   {
14     public DelphiSqlClientException() : base() { }
15     public DelphiSqlClientException(string Message) : base(Message) { }
16     public DelphiSqlClientException(string Message, Exception
           InnerException) : base(Message, InnerException) { }
17   }
18 }
```

A.2.3 SQL Namespace

Two files comprise the SQL namespace. The SQLSettings class, which enables a custom configuration element for storing the runtime settings data for the SQLClient library, and the SQLClient

library itself, which contains the methods that manipulate the underlying SQL database. It should be noted that the SQL methods in use are all strongly typed for Microsoft SQL Server. A different ADO.NET provider would be necessary for these methods to be used with a different database provider.

Listing 36: SQL Client Custom Settings Class

```
1  //
2  // SQL Configuration section:
3  // Provides a uniform method with which to retrieve the SQL connection
       string:
4  // (C) 2014 <gg_duncan@laurentian.ca>
5  //
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Configuration;
11
12 namespace Delphi.Data.SQL
13 {
14   public class SQLSettings : ConfigurationSection
15   {
16     public const string CONFIG_SETTINGS_ELEMENT_NAME = "SQLSettings";
17
18     private static SQLSettings _settings = ConfigurationManager.
         GetSection(CONFIG_SETTINGS_ELEMENT_NAME) as SQLSettings;
19     public static SQLSettings Settings { get { return _settings; } }
20
```

```
21      [ConfigurationProperty("ConnectionString", DefaultValue = "",
            IsRequired = true)]
22      [StringValidator()]
23      public string ConnectionString
24      {
25        get { return this["ConnectionString"].ToString(); }
26        set { this["ConnectionString"] = value; }
27      }
28    }
29  }
```

Listing 37: SQL Client Class

```
1  //
2  // SqlClient.cs
3  // Represents a SQL client connection to the DelphiPC system data
4  // (c) 2014 Grant G. O. Duncan <gg_duncan@laurentian.ca>
5  //
6  using System;
7  using System.Collections.Generic;
8  using System.Data;
9  using System.Data.SqlClient;
10 using System.Linq;
11 using System.Text;
12 using Delphi.Data.Entities;
13 using Delphi.Data.Exceptions;
14
15 namespace Delphi.Data.SQL
16 {
17   public class SqlClient
```

```
18  {
19      #region Constants:
20      //
21      // "Get" SQL Procs:
22      //
23      private const string SP_Get_Session_by_Session_ID = "
            Get_Session_by_Session_ID";
24      private const string SP_Get_User_by_User_ID = "Get_User_By_User_ID"
            ;
25      private const string SP_Get_Rounds_By_Session_ID = "
            Get_Rounds_By_Session_ID";
26      private const string SP_Get_Ratings_By_Round_ID = "
            Get_Ratings_By_Round_ID";
27      private const string SP_Get_Criteria_By_Criteria_ID = "
            Get_Criteria_By_Criteria_ID";
28      private const string SP_Get_Range_By_Range_ID = "
            Get_Range_By_Range_ID";
29      private const string SP_Get_Crtieria_By_Session_ID = "
            Get_Criteria_By_Session_ID";
30      private const string SP_Get_User_By_GUID = "Get_User_by_GUID";
31      private const string SP_Get_User_Rounds_By_GUID = "
            Resolve_User_Rounds_by_GUID";
32      private const string SP_Get_Last_Round_By_Session_ID = "
            Get_Last_Round_by_Session_ID";
33      //
34      // "Add/Update" SQL Procs:
35      //
36      private const string SP_Add_Update_Session = "Add_Update_Session";
```

```
37      private const string SP_Add_Update_Criteria = "Add_Update_Criteria"
            ;
38      private const string SP_Add_Update_Range = "Add_Update_Range";
39      private const string SP_Add_Update_Rating = "Add_Update_Rating";
40      private const string SP_Add_Update_Round = "Add_Update_Round";
41      private const string SP_Add_Update_User = "Add_Update_User";
42      private const string SP_Set_Consent = "Set_Consent";
43      private const string SP_End_Session = "EndSession";
44      private const string SP_End_Round = "EndRound";
45      #endregion
46      #region Public Properties:
47      // Connection string:
48      public string ConnectionString { get; private set; }
49      #endregion
50      #region .ctor:
51      // Default constructor: grabs the connection string from the
            settings element within the application's
52      // run-time configuration file.
53      public SqlClient()
54      {
55        ConnectionString = Delphi.Data.SQL.SQLSettings.Settings.
              ConnectionString;
56        if (ConnectionString == null || ConnectionString == "") throw new
              NullReferenceException("Connection String is null.");
57      }
58      // Constructor that takes a connection string (overriding any run-
            time configuration):
59      public SqlClient(string ConnectionString)
60      {
```

```
61          if (ConnectionString == null || ConnectionString == "") throw new
                NullReferenceException("Connection String is null.");
62          this.ConnectionString = ConnectionString;
63      }
64      #endregion
65      #region Add/Update Methods:
66
67      public void New_Round(Round round)
68      {
69          using (SqlConnection con = new SqlConnection(ConnectionString))
70          {
71              try
72              {
73                  DateTime EndDtTm = DateTime.Now;
74                  SqlCommand com = new SqlCommand(SP_End_Round, con);
75                  com.CommandType = CommandType.StoredProcedure;
76                  com.Parameters.AddWithValue("@Round_ID", round.Round_ID);
77                  con.Open();
78                  com.ExecuteNonQuery();
79                  con.Close();
80              }
81              finally
82              {
83                  if (con.State != ConnectionState.Closed)
84                      con.Close();
85              }
86          }
87      }
88
```

```
89      public void End_Session(DelphiSession session, Round round)
90      {
91        using (SqlConnection con = new SqlConnection(ConnectionString))
92        {
93          try
94          {
95            DateTime EndDtTm = DateTime.Now;
96            SqlCommand com = new SqlCommand(SP_End_Session, con);
97            com.CommandType = CommandType.StoredProcedure;
98            com.Parameters.AddWithValue("@Session_ID", session.Session_ID
                  );
99            com.Parameters.AddWithValue("@Round_ID", round.Round_ID);
100           com.Parameters.AddWithValue("@End_DtTm", EndDtTm);
101           con.Open();
102           com.ExecuteNonQuery();
103           con.Close();
104           session.Ended_DtTm = EndDtTm;
105           round.End_DtTm = EndDtTm;
106         }
107         finally
108         {
109           if (con.State != ConnectionState.Closed)
110             con.Close();
111         }
112       }
113     }
114
115     public void Set_Consent(Entities.User UserInfo)
116     {
```

```
117         using (SqlConnection con = new SqlConnection(ConnectionString))
118         {
119           try
120           {
121             SqlCommand com = new SqlCommand(SP_Set_Consent, con);
122             com.CommandType = CommandType.StoredProcedure;
123             com.Parameters.AddWithValue("@GUID", UserInfo.GUID);
124             con.Open();
125             com.ExecuteNonQuery();
126             con.Close();
127           }
128           finally
129           {
130             if (con.State != ConnectionState.Closed)
131               con.Close();
132           }
133         }
134       }
135
136       public void Add_Update_DelphiSession(ref DelphiSession DSession)
137       {
138         using (SqlConnection con = new SqlConnection(ConnectionString))
139         {
140           try
141           {
142             SqlCommand com = new SqlCommand(SP_Add_Update_Session, con);
143             com.CommandType = CommandType.StoredProcedure;
144             com.Parameters.AddWithValue("@Create_DtTm", DSession.
                  Create_DtTm);
```

```
145             com.Parameters.AddWithValue("@Round_Threshold", DSession.
                    Round_Threshold);
146             com.Parameters.AddWithValue("@Description", DSession.
                    Description);
147             com.Parameters.AddWithValue("@Short_Description", DSession.
                    Short_Description);
148             com.Parameters.AddWithValue("@Started_DtTm", DSession.
                    Started_DtTm);
149             com.Parameters.AddWithValue("@Ended_DtTm", DSession.
                    Ended_DtTm);
150
151             SqlParameter ParamSession_Id = new SqlParameter("@Session_ID"
                    , SqlDbType.Int);
152             ParamSession_Id.Direction = ParameterDirection.InputOutput;
153             ParamSession_Id.Value = DSession.Session_ID;
154
155             com.Parameters.Add(ParamSession_Id);
156             con.Open();
157             com.ExecuteNonQuery();
158             if (DSession.Session_ID != (int)ParamSession_Id.Value)
                    DSession.Session_ID = (int)ParamSession_Id.Value;
159
160             Add_Update_Rounds(DSession.Rounds, DSession.Session_ID, con,
                    false);
161
162             con.Close();
163         }
164         catch (Exception e)
165         {
```

```
166             throw new DelphiSqlClientException("Add_Update_DelphiSession␣
                    Exception", e);
167           }
168           finally
169           {
170             if (con.State != ConnectionState.Closed) con.Close();
171           }
172         }
173       }
174
175       public void Add_Update_Range(ref Range range, SqlConnection con,
              bool CloseConnection)
176       {
177         SqlCommand com = new SqlCommand(SP_Add_Update_Range, con);
178         com.CommandType = CommandType.StoredProcedure;
179
180         SqlParameter ParamRange_ID = new SqlParameter("@Range_ID",
                SqlDbType.Int);
181         ParamRange_ID.Direction = ParameterDirection.InputOutput;
182         SqlParameter ParamHigh_Range = new SqlParameter("@High_Range",
                SqlDbType.Decimal);
183         SqlParameter ParamLow_Range = new SqlParameter("@Low_Range",
                SqlDbType.Decimal);
184         SqlParameter ParamStep = new SqlParameter("@Step", SqlDbType.
                Decimal);
185
186         ParamRange_ID.Value = range.Range_ID; ;
187         ParamHigh_Range.Value = range.High_Range;
188         ParamLow_Range.Value = range.Low_Range;
```

```
189         ParamStep.Value = range.Step;
190         com.Parameters.AddRange(new SqlParameter[] { ParamRange_ID,
                ParamHigh_Range, ParamLow_Range, ParamStep });
191
192         if (con.State != ConnectionState.Open) con.Open();
193
194         com.ExecuteNonQuery();
195         range.Range_ID = (int)ParamRange_ID.Value;
196
197         if (CloseConnection) con.Close();
198       }
199
200       public void Add_Update_Criteria(ref Criteria criteria,
             SqlConnection con, bool CloseConnection)
201       {
202         SqlCommand com = new SqlCommand(SP_Add_Update_Criteria, con);
203         com.CommandType = CommandType.StoredProcedure;
204
205         SqlParameter ParamDescription = new SqlParameter("@Description",
                SqlDbType.NVarChar);
206         SqlParameter ParamShort_Description = new SqlParameter("
                @Short_Description", SqlDbType.NVarChar);
207         SqlParameter ParamCreate_DtTm = new SqlParameter("@Create_DtTm",
                SqlDbType.DateTime);
208         SqlParameter ParamCreated_By_User_ID = new SqlParameter("
                @Created_By_User_ID", SqlDbType.Int);
209         SqlParameter ParamRange_ID = new SqlParameter("@Range_ID",
                SqlDbType.Int);
```

```
210         SqlParameter ParamCriteria_ID = new SqlParameter("@Criteria_ID",
                SqlDbType.Int);
211         ParamCriteria_ID.Direction = ParameterDirection.InputOutput;
212
213         com.Parameters.AddRange(new SqlParameter[] { ParamDescription,
                ParamShort_Description, ParamCreate_DtTm,
214           ParamCreated_By_User_ID, ParamRange_ID, ParamCriteria_ID });
215
216         if (con.State != ConnectionState.Open) con.Open();
217
218         ParamDescription.Value = criteria.Description;
219         ParamShort_Description.Value = criteria.Short_Description;
220         ParamCreate_DtTm.Value = criteria.Create_DtTm;
221         ParamCreated_By_User_ID.Value = criteria.CreatedBy.User_ID;
222         ParamRange_ID.Value = criteria.CriteriaRange.Range_ID;
223         ParamCriteria_ID.Value = criteria.Criteria_ID;
224
225         com.ExecuteNonQuery();
226
227         criteria.Criteria_ID = (int)ParamCriteria_ID.Value;
228
229         if (CloseConnection) con.Close();
230       }
231
232       public void Add_Update_Ratings(List<Rating> Ratings, int Round_ID)
233       {
234         using (SqlConnection con = new SqlConnection(ConnectionString))
235         {
236           try
```

```
237         {
238             SqlCommand com = new SqlCommand(SP_Add_Update_Rating, con);
239             com.CommandType = CommandType.StoredProcedure;
240             SqlParameter ParamRating_ID = new SqlParameter("@Rating_ID",
                    SqlDbType.Int);
241             ParamRating_ID.Direction = ParameterDirection.InputOutput;
242             SqlParameter ParamRound_ID = new SqlParameter("@Round_ID",
                    SqlDbType.Int);
243             SqlParameter ParamLHS_Criteria_ID = new SqlParameter("
                    @LHS_Criteria_ID", SqlDbType.Int);
244             SqlParameter ParamLHS_Criteria_Value = new SqlParameter("
                    @LHS_Criteria_Value", SqlDbType.Decimal);
245             SqlParameter ParamRHS_Criteria_ID = new SqlParameter("
                    @RHS_Criteria_ID", SqlDbType.Int);
246             SqlParameter ParamRHS_Criteria_Value = new SqlParameter("
                    @RHS_Criteria_Value", SqlDbType.Decimal);
247             SqlParameter ParamRated_DtTm = new SqlParameter("@Rated_DtTm"
                    , SqlDbType.DateTime);
248             SqlParameter ParamRated_By_User_ID = new SqlParameter("
                    @Rated_By_User_ID", SqlDbType.Int);
249             SqlParameter ParamSlider_Pos = new SqlParameter("@Slider_Pos"
                    , SqlDbType.Int);
250
251             com.Parameters.AddRange(new SqlParameter[]{ ParamRating_ID,
                    ParamRound_ID, ParamLHS_Criteria_ID,
252                 ParamLHS_Criteria_Value, ParamRHS_Criteria_ID,
                        ParamRHS_Criteria_Value, ParamRated_DtTm,
253                 ParamRated_By_User_ID, ParamSlider_Pos});
254
```

```
255             con.Open();
256             foreach (Rating rating in Ratings)
257             {
258               ParamRating_ID.Value = rating.Rating_ID;
259               ParamRound_ID.Value = Round_ID;
260
261               ParamLHS_Criteria_ID.Value = rating.LHS_Criteria.
                      Criteria_ID;
262               ParamLHS_Criteria_Value.Value = rating.LHS_Criteria_Value;
263               ParamRHS_Criteria_ID.Value = rating.RHS_Criteria.
                      Criteria_ID;
264               ParamRHS_Criteria_Value.Value = rating.RHS_Criteria_Value;
265               ParamRated_DtTm.Value = rating.Rated_DtTm;
266               ParamRated_By_User_ID.Value = rating.Rated_By.User_ID;
267               ParamSlider_Pos.Value = rating.Slider_Pos;
268               com.ExecuteNonQuery();
269               rating.Rating_ID = (int)ParamRating_ID.Value;
270             }
271             con.Close();
272           }
273           finally
274           {
275             if (con.State != ConnectionState.Closed) con.Close();
276           }
277         }
278       }
279
280       public void Add_Update_Rounds(List<Round> Rounds, int Session_ID,
            SqlConnection con, bool CloseConnection)
```

```
281     {
282         SqlCommand com = new SqlCommand(SP_Add_Update_Round, con);
283         com.CommandType = CommandType.StoredProcedure;
284         SqlParameter ParamStart_DtTm = new SqlParameter("@Start_DtTm",
                SqlDbType.DateTime);
285         SqlParameter ParamEnd_DtTm = new SqlParameter("@End_DtTm",
                SqlDbType.DateTime);
286         SqlParameter ParamRound_Number = new SqlParameter("@Round_Number"
                , SqlDbType.Int);
287         SqlParameter ParamSession_ID = new SqlParameter("@Session_ID",
                SqlDbType.Int);
288         SqlParameter ParamRound_ID = new SqlParameter("@Round_ID",
                SqlDbType.Int);
289         ParamRound_ID.Direction = ParameterDirection.InputOutput;
290         com.Parameters.AddRange(new SqlParameter[] { ParamSession_ID,
                ParamStart_DtTm, ParamEnd_DtTm, ParamRound_ID, ParamRound_ID
                });
291
292         if (con.State != ConnectionState.Open) con.Open();
293
294         foreach (Round r in Rounds)
295         {
296             ParamStart_DtTm.Value = r.Start_DtTm;
297             ParamEnd_DtTm.Value = r.End_DtTm;
298             ParamRound_Number.Value = r.Round_Number;
299             ParamSession_ID.Value = Session_ID;
300             ParamRound_ID.Value = r.Round_ID;
301             com.ExecuteNonQuery();
302             r.Round_ID = (int)ParamRound_ID.Value;
```

```
303         }
304
305         if (CloseConnection) con.Close();
306
307     }
308     #endregion
309     #region Get/Retrieval Methods:
310
311     public Tuple<Round, DelphiSession, User> Get_User_Rounds_by_GUID(
            string UserGUID)
312     {
313       Round r = null; DelphiSession s = null; User u = null;
314       using (SqlConnection con = new SqlConnection(ConnectionString))
315       {
316         try
317         {
318           SqlCommand com = new SqlCommand(SP_Get_User_Rounds_By_GUID,
                  con);
319           com.CommandType = CommandType.StoredProcedure;
320           com.Parameters.AddWithValue("@UserGUID", UserGUID);
321           con.Open();
322           SqlDataReader reader = com.ExecuteReader(CommandBehavior.
                  CloseConnection);
323           if (reader.Read())
324           {
325             s = BuildSession(reader);
326             u = BuildUser(reader);
327             r = BuildRound(reader);
328             // Can we continue?
```

```
329             r.CanProceed = reader["CanProceed"] != DBNull.Value &&
                    reader["CanProceed"].ToString() == "1";
330             r.Session_Is_Finished = reader["Session_is_finished"] !=
                    DBNull.Value && reader["Session_is_finished"].ToString()
                     == "1";
331           }
332           reader.Close();
333         }
334         finally
335         {
336           if (con.State != ConnectionState.Closed)
337             con.Close();
338         }
339       }
340       return new Tuple<Round, DelphiSession, User>(r, s, u);
341     }
342
343
344     public DelphiSession BuildSession(SqlDataReader r)
345     {
346       DelphiSession retval = new DelphiSession();
347       retval.Create_DtTm = Convert.ToDateTime(r["Create_DtTm"]);
348       retval.Description = r["Description"].Equals(DBNull.Value) ? "" :
              r["Description"].ToString();
349       retval.Ended_DtTm = r["Ended_DtTm"].Equals(DBNull.Value) ?
             DateTime.MinValue : Convert.ToDateTime(r["Ended_DtTm"]);
350       retval.Round_Threshold = Convert.ToInt32(r["Round_Threshold"]);
351       retval.Session_ID = Convert.ToInt32(r["Session_ID"]);
```

```
352         retval.Short_Description = r["Short_Description"].Equals(DBNull.
                Value) ? "" : r["Short_Description"].ToString();
353         retval.Started_DtTm = r["Started_DtTm"].Equals(DBNull.Value) ?
                DateTime.MinValue : Convert.ToDateTime(r["Started_DtTm"]);
354         return retval;
355       }
356
357       public List<Tuple<int, int, int>> GetLastRoundBySessionID(int
              Session_ID)
358       {
359         List<Tuple<int, int, int>> retval = new List<Tuple<int, int, int
                >>();
360         using (SqlConnection con = new SqlConnection(ConnectionString))
361         {
362           try
363           {
364             SqlCommand com = new SqlCommand(
                    SP_Get_Last_Round_By_Session_ID, con);
365             com.CommandType = CommandType.StoredProcedure;
366             com.Parameters.AddWithValue("@Session_ID", Session_ID);
367             con.Open();
368             SqlDataReader r = com.ExecuteReader(CommandBehavior.
                    CloseConnection);
369             while (r.Read())
370             {
371               retval.Add(new Tuple<int, int, int>(Convert.ToInt32(r["
                      LHS_Criteria_id"]),
372                 Convert.ToInt32(r["RHS_Criteria_ID"]), Convert.ToInt32(r[
                        "avg_slider_pos"])));
```

```
373             }
374             r.Close();
375         }
376         finally
377         {
378             if (con.State != ConnectionState.Closed)
379                 con.Close();
380         }
381     }
382     return retval;
383 }
384
385 public List<Rating> GetRatingsByRoundID(int Round_ID)
386 {
387     List<Rating> retval = new List<Rating>();
388     Rating rating = null;
389     using (SqlConnection con = new SqlConnection(ConnectionString))
390     {
391         try
392         {
393             SqlCommand com = new SqlCommand(SP_Get_Ratings_By_Round_ID,
                    con);
394             com.CommandType = CommandType.StoredProcedure;
395             com.Parameters.AddWithValue("@Round_ID", Round_ID);
396             con.Open();
397             SqlDataReader r = com.ExecuteReader(CommandBehavior.
                    CloseConnection);
398             while (r.Read())
399             {
```

```
400             rating = new Rating();
401             if (Rating.NumCriteria <= 0) Rating.NumCriteria = Convert.
                    ToInt32(r["criteria_count"]);
402             rating.Rated_By = BuildUserByUserID(Convert.ToInt32(r["
                    Rated_By_User_ID"]));
403             rating.Rated_DtTm = Convert.ToDateTime(r["Rated_DtTm"]);
404             rating.LHS_Criteria_Value = Convert.ToDecimal(r["
                    LHS_Criteria_Value"]);
405             rating.RHS_Criteria_Value = Convert.ToDecimal(r["
                    RHS_Criteria_Value"]);
406             rating.Rating_ID = Convert.ToInt32(r["Rating_ID"]);
407             rating.Slider_Pos = Convert.ToInt32(r["Slider_Pos"]);
408             retval.Add(rating);
409           }
410           r.Close();
411         }
412         finally
413         {
414           if (con.State != ConnectionState.Closed)
415             con.Close();
416         }
417       }
418       return retval;
419     }
420
421
422     public List<Criteria> GetCriteriaBySessionID(int Session_ID)
423     {
424       List<Criteria> retval = null;
```

```
425         using (SqlConnection con = new SqlConnection(ConnectionString))
426         {
427           try
428           {
429             retval = GetCriteriaBySessionID(con, Session_ID, true);
430           }
431           finally
432           {
433             if (con.State != ConnectionState.Closed) con.Close();
434           }
435         }
436         return retval;
437       }
438
439       private List<Criteria> GetCriteriaBySessionID(SqlConnection con,
              int Session_ID, bool CloseConnection)
440       {
441         List<Criteria> retval = new List<Criteria>();
442         Criteria c = null;
443
444         SqlCommand com = new SqlCommand(SP_Get_Crtieria_By_Session_ID,
                con);
445         com.CommandType = CommandType.StoredProcedure;
446         com.Parameters.AddWithValue("@Session_ID", Session_ID);
447
448         if (con.State != ConnectionState.Open) con.Open();
449
450         SqlDataReader r = com.ExecuteReader(CommandBehavior.SingleResult
              | (CloseConnection ? CommandBehavior.CloseConnection :
```

```
            CommandBehavior.Default));
451
452         while (r.Read())
453         {
454           c = new Criteria();
455           c.Create_DtTm = Convert.ToDateTime(r["Create_DtTm"]);
456           c.CreatedBy = BuildUserByUserID(Convert.ToInt32(r["
                Created_By_User_ID"]));
457           c.Criteria_ID = Convert.ToInt32(r["Criteria_ID"]);
458           c.Description = r["Description"].ToString();
459           c.Short_Description = r["Short_Description"].ToString();
460           c.CriteriaRange = BuildRange(Convert.ToInt32(r["Range_ID"]));
461           retval.Add(c);
462         }
463
464         r.Close();
465
466         return retval;
467       }
468
469       public Criteria BuildCriteria(SqlConnection con, int Criteria_ID,
            bool CloseConnection)
470       {
471         Criteria retval = null;
472
473         SqlCommand com = new SqlCommand(SP_Get_User_by_User_ID, con);
474         com.CommandType = CommandType.StoredProcedure;
475         com.Parameters.AddWithValue("@Criteria_ID", Criteria_ID);
476
```

```
477         if (con.State != ConnectionState.Open) con.Open();
478
479         SqlDataReader r = com.ExecuteReader(CommandBehavior.SingleResult
                | (CloseConnection ? CommandBehavior.CloseConnection :
                CommandBehavior.Default));
480
481         if (r.Read())
482         {
483           retval = new Criteria();
484           retval.Create_DtTm = Convert.ToDateTime(r["Create_DtTm"]);
485           retval.CreatedBy = BuildUserByUserID(Convert.ToInt32(r["
                Created_By_User_ID"]));
486           retval.Criteria_ID = Criteria_ID;
487           retval.Description = r["Description"].ToString();
488           retval.Short_Description = r["Short_Description"].ToString();
489           retval.CriteriaRange = BuildRange(Convert.ToInt32(r["Range_ID"
                ]));
490         }
491
492         r.Close();
493
494         return retval;
495       }
496
497       private Range BuildRange(int Range_ID)
498       {
499         Range retval = null;
500
501         using (SqlConnection con = new SqlConnection(ConnectionString))
```

```
502         {
503             SqlCommand com = new SqlCommand(SP_Get_Range_By_Range_ID, con);
504             com.CommandType = CommandType.StoredProcedure;
505             com.Parameters.AddWithValue("@Range_ID", Range_ID);
506 
507             if (con.State != ConnectionState.Open) con.Open();
508 
509             SqlDataReader r = com.ExecuteReader(CommandBehavior.
                    SingleResult | CommandBehavior.CloseConnection );
510 
511             if (r.Read())
512             {
513                 retval = new Range();
514                 retval.High_Range = Convert.ToDouble(r["High_Range"]);
515                 retval.Low_Range = Convert.ToDouble(r["Low_Range"]);
516                 retval.Step = Convert.ToDouble(r["Step"]);
517                 retval.Range_ID = Range_ID;
518             }
519             r.Close();
520         }
521 
522         return retval;
523     }
524 
525     public User GetUserByGUID(string UserGUID)
526     {
527         User retval = new User();
528 
529         using (SqlConnection con = new SqlConnection(ConnectionString))
```

```
530         {
531             SqlCommand com = new SqlCommand(SP_Get_User_By_GUID, con);
532             com.CommandType = CommandType.StoredProcedure;
533             com.Parameters.AddWithValue("@GUID", UserGUID);
534 
535             if (con.State != ConnectionState.Open) con.Open();
536 
537             SqlDataReader r = com.ExecuteReader(CommandBehavior.
                    SingleResult | CommandBehavior.CloseConnection);
538             if(r.Read()) retval = BuildUser(r);
539             r.Close();
540         }
541 
542         return retval;
543 
544     }
545 
546     public User BuildUserByUserID(int User_ID)
547     {
548         User retval = new User();
549 
550         using (SqlConnection con = new SqlConnection(ConnectionString))
551         {
552             SqlCommand com = new SqlCommand(SP_Get_User_by_User_ID, con);
553             com.CommandType = CommandType.StoredProcedure;
554             com.Parameters.AddWithValue("@User_ID", User_ID);
555 
556             if (con.State != ConnectionState.Open) con.Open();
557 
```

```
558         SqlDataReader r = com.ExecuteReader(CommandBehavior.
                SingleResult | CommandBehavior.CloseConnection);
559         if(r.Read()) retval = BuildUser(r);
560         r.Close();
561       }
562
563       return retval;
564     }
565
566     private User BuildUser(SqlDataReader r)
567     {
568       User retval = new User();
569       retval.User_ID = Convert.ToInt32(r["User_ID"]);
570       retval.GUID = r["GUID"].ToString();
571       retval.HasConsented = Convert.ToBoolean(r["Has_Consented"].Equals
            (DBNull.Value) ? false : r["Has_Consented"]);
572       if (retval.HasConsented)
573         retval.Consent_DtTm = Convert.ToDateTime(r["Consent_DtTm"]);
574       retval.IsAdministrator = r["Is_Administrator"].Equals(DBNull.
            Value) ? false : Convert.ToBoolean(r["Is_Administrator"]);
575       return retval;
576     }
577
578     public Round BuildRound(SqlDataReader r)
579     {
580       Round round = new Round();
581       round.End_DtTm = r["End_DtTm"].Equals(DBNull.Value) ? DateTime.
            MinValue : Convert.ToDateTime(r["End_DtTm"]);
582       round.Start_DtTm = Convert.ToDateTime(r["Start_DtTm"]);
```

```
583             round.Round_Number = Convert.ToInt32(r["Round_Number"]);
584             round.Round_ID = Convert.ToInt32(r["Round_ID"]);
585
586             return round;
587         }
588         #endregion
589     }
590 }
```

A.3 PC Source Code Listing

The following module implements the pairwise comparison mathematical routines. These methods are implemented as static methods, therefore must be called by the full specification of class and method (for instance: PCMath.Averages()).

Listing 38: PC Math

```
1  //
2  // Static methods used for Generating PC values from their
3  // Principal generators
4  //
5  // <gg_duncan@laurentian.ca>
6  //
7  using System;
8  using System.Collections.Generic;
9  using System.Linq;
10 using System.Text;
11
12 namespace PC
13 {
```

```
14    public static class PCMath
15    {
16      //
17      // Builds a PC matrix from an array of comparisons:
18      //
19      public static decimal[,] Build_PC_Mat(decimal[] C, int n)
20      {
21        decimal[,] A = new decimal[n, n];
22        int k = 0;
23        for (int i = 0; i < n; i++)
24        {
25          for (int j = 0; j < n; j++)
26          {
27            if (i == j) A[i, j] = 1m;
28            else if (i > j) A[i, j] = 1.00m / A[j, i];
29            else if (i < j) A[i, j] = C[k++];
30          }
31        }
32        return A;
33      }
34      //
35      // Calculates an array that is the average for each element
36      // in the list (row-wise over a 1 dimensional matrix)
37      //
38      public static decimal[] Averages (List<decimal[]> list)
39      {
40        decimal[] averages = null;
41        if(list.Count > 0)
42        {
```

```
43          averages = new decimal[list[0].Length];
44          for (int i = 0; i < averages.Length; i++) averages[i] = 0;
45          foreach (decimal[] listitem in list)
46          {
47            for (int i = 0; i < listitem.Length; i++)
48            {
49              averages[i] += listitem[i]/list.Count;
50            }
51          }
52        }
53        return averages;
54      }
55      //
56      // Calculates the Geometric mean for each row in the given matrix,
57      // returns an array of size N for N rows of the matrix:
58      //
59      public static decimal[] Geometric_Means(decimal[,] A)
60      {
61        int n = A.GetLength(0);
62        int m = A.GetLength(1);
63        decimal[] M = new decimal[n];
64
65        for (int i = 0; i < n; i++)
66        {
67          M[i] = 1m;
68          for (int j = 0; j < m; j++)
69          {
70            M[i] *= A[i, j];
71          }
```

```
72          M[i] = (decimal)Math.Pow((double)M[i], 1.0 / m);
73        }
74        return M;
75      }
76
77      //
78      // Normalizes the Items in a decimal array:
79      //
80      public static decimal[] Normalize(decimal[] A)
81      {
82        int n = A.Length;
83        decimal total = 0m;
84        foreach (decimal d in A) total += d;
85        decimal[] M = new decimal[n];
86        for (int i = 0; i < n; i++) M[i] = (A[i] / total) * 100m;
87        return M;
88      }
89
90      //
91      // Calculates Koczkodaj's inconsistency indicator of a matrix A
92      //
93      public static decimal Calculate_ii(decimal[,] A, out int outi, out
            int outj, out int outk)
94      {
95        decimal ii = 0, ii_t = 0;
96        int k = 0, n = A.GetLength(0);
97        outi = outj = outk = 0;
98
99        for (int j = n-1; j >= 0; j--)
```

```
100         {
101           k = j - 1;
102           for (int i = j - 2; i >= 0; i--)
103           {
104             ii_t = Math.Min(
105               Math.Abs(1.0m - (A[i, j] / A[i, k] * A[k, j])),
106               Math.Abs(1.0m - (A[i, k] * A[k, j] / A[i, j]))
107             );
108             if (ii_t > ii)
109             {
110               ii = ii_t;
111               outi = i; outj = j; outk = k;
112             }
113           }
114           k--;
115         }
116         return ii;
117       }
118       //
119       // Modify the referenced double[,] by creating it's generatable PC
            values
120       // Add inconsistency to the specified triad
121       // Thread-safe
122       //
123       private static Object Lock = new object();
124       public static void GenerateFromPrincipalGeneretors(ref decimal[,] A
            , decimal DefaultInconsistency, int Triad)
125       {
126         int n = A.GetLength(0);
```

```
127         int k = 0, iteration = 0;
128         for (int j = n; j > 0; j--)
129         {
130           k = j-1;
131           for (int i = j - 2; i > 0; i--)
132           {
133             lock(Lock)
134             {
135               A[i, j] = A[i, k] * A[k, j] * (iteration == Triad ? 1.00m
                      :(1.00m + DefaultInconsistency));
136             }
137             k--;
138           }
139         }
140       }
141     }
142 }
```

B PC Delphi SQL Database Listings

The following section lists the TSQL database source code and entity definitions, including the TSQL Stored Procedure listings. Note that permission sets are not included in this listing.

B.1 SQL Database Tables

The following source code listings contain the definitions for the PC Delphi implementation's SQL database tables/entities (8 in total).

Listing 39: Ratings.sql

```
1  CREATE TABLE [dbo].[Ratings] (
2      [Rating_ID]          INT              IDENTITY (1, 1) NOT NULL,
3      [Round_ID]           INT              NOT NULL,
4      [LHS_Criteria_ID]    INT              NULL,
5      [LHS_Criteria_Value] DECIMAL (18, 5)  NULL,
6      [RHS_Criteria_ID]    INT              NULL,
7      [RHS_Criteria_Value] DECIMAL (18, 5)  NULL,
8      [Rated_DtTm]         DATETIME         NULL,
9      [Rated_By_User_ID]   INT              NOT NULL,
10     [Slider_Pos]         INT              NULL,
11     CONSTRAINT [PK_Ratings] PRIMARY KEY CLUSTERED ([Rating_ID] ASC)
12 );
```

Listing 40: Users.sql

```
1  CREATE TABLE [dbo].[Users] (
2      [User_ID]            INT              IDENTITY (1, 1) NOT NULL,
3      [GUID]               NVARCHAR (256)   NOT NULL,
4      [Has_Consented]      BIT              NULL,
5      [Consent_DtTm]       DATETIME         NULL,
```

```
6       [Is_Administrator] BIT                 NULL,
7       CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED ([User_ID] ASC, [GUID]
            ASC)
8   );
```

Listing 41: Criteria.sql

```
1   CREATE TABLE [dbo].[Criteria] (
2       [Criteria_ID]          INT                IDENTITY (1, 1) NOT NULL,
3       [Description]          NVARCHAR (1024)    NULL,
4       [Short_Description]    NVARCHAR (50)      NULL,
5       [Create_DtTm]          DATETIME           NULL,
6       [Created_By_User_ID]   INT                NOT NULL,
7       [Range_ID]             INT                NOT NULL,
8       CONSTRAINT [PK_Criteria] PRIMARY KEY CLUSTERED ([Criteria_ID] ASC)
9   );
```

Listing 42: Range.sql

```
1   CREATE TABLE [dbo].[Range] (
2       [Range_ID]    INT                  IDENTITY (1, 1) NOT NULL,
3       [High_Range]  DECIMAL (18, 3)      NOT NULL,
4       [Low_Range]   DECIMAL (18, 3)      NOT NULL,
5       [Step]        DECIMAL (18, 3)      NOT NULL,
6       CONSTRAINT [PK_Range] PRIMARY KEY CLUSTERED ([Range_ID] ASC)
7   );
```

Listing 43: UsersRounds.sql

```
1   CREATE TABLE [dbo].[Users_Rounds] (
2       [User_ID]   INT NOT NULL,
3       [Round_ID]  INT NOT NULL,
```

```
4       CONSTRAINT [PK_Users_Rounds] PRIMARY KEY CLUSTERED ([User_ID] ASC,
            [Round_ID] ASC)
5   );
```

Listing 44: SessionCriteria.sql

```
1   CREATE TABLE [dbo].[Session_Criteria] (
2       [Session_ID]   INT NOT NULL,
3       [Criteria_ID] INT NOT NULL,
4       CONSTRAINT [PK_Session_Criteria] PRIMARY KEY CLUSTERED ([Session_ID
            ] ASC, [Criteria_ID] ASC)
5   );
```

Listing 45: Rounds.sql

```
1   CREATE TABLE [dbo].[Rounds] (
2       [Round_ID]      INT      IDENTITY (1, 1) NOT NULL,
3       [Start_DtTm]    DATETIME NOT NULL,
4       [End_DtTm]      DATETIME NULL,
5       [Round_Number] INT      NULL,
6       [Session_ID]    INT      NULL,
7       CONSTRAINT [PK_Rounds] PRIMARY KEY CLUSTERED ([Round_ID] ASC)
8   );
```

Listing 46: Sessions.sql

```
1   CREATE TABLE [dbo].[Sessions] (
2       [Session_ID]          INT                IDENTITY (1, 1) NOT NULL,
3       [Create_DtTm]         DATETIME           NOT NULL,
4       [Round_Threshold]     INT                NOT NULL,
5       [Created_By_User_ID] INT                NOT NULL,
6       [Description]         NVARCHAR (1024)    NULL,
7       [Short_Description]   NVARCHAR (50)      NULL,
```

```
8       [Started_DtTm]          DATETIME        NULL,
9       [Ended_DtTm]            DATETIME        NULL,
10      CONSTRAINT [PK_Sessions] PRIMARY KEY CLUSTERED ([Session_ID] ASC)
11  );
```

B.2 Stored Procedures

The following source code listings contain the definitions for the PC Delphi implementation's TSQL stored procedures (20 in total).

Listing 47: Get Last Round by Session ID

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Get last round by session ID
5  -- ============================================
6  CREATE PROCEDURE [dbo].[Get_Last_Round_by_Session_ID]
7    -- Add the parameters for the stored procedure here
8    @Session_ID int
9  AS
10 BEGIN
11   -- SET NOCOUNT ON added to prevent extra result sets from
12   -- interfering with SELECT statements.
13   SET NOCOUNT ON;
14   DECLARE @Round_ID int
15   DECLARE @Round_Number int
16
17   SET @Round_Number = (SELECT TOP 1 max(round_number) from Rounds where
         session_id = @Session_ID)
```

```
18    SET @Round_ID = (Select top 1 Round_ID from Rounds WHERE Session_ID =
          @Session_ID and Round_number
19     = @Round_number -1)
20
21    select
22      LHS_Criteria_id,
23      RHS_Criteria_id,
24      round(avg(cast(slider_pos as decimal(12,2))),0) as avg_slider_pos
25    from
26      ratings r
27    WHERE
28      r.Round_ID = @Round_ID
29    group by
30      LHS_Criteria_id, RHS_Criteria_id
31    order by
32      LHS_Criteria_Id, RHS_Criteria_id
33  END
```

Listing 48: Add and Update Criteria

```
1  -- =============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Adds or Updates the Criteria information
5  -- =============================================
6  CREATE PROCEDURE [dbo].[Add_Update_Criteria]
7
8  @Description nvarchar(1024),
9  @Short_Description nvarchar(50),
10 @Create_DtTm datetime,
```

```
11  @Created_By_User_ID int,
12  @Range_ID int,
13  @Criteria_ID int OUTPUT
14
15  AS
16
17  BEGIN
18    -- SET NOCOUNT ON added to prevent extra result sets from
19    -- interfering with SELECT statements.
20    SET NOCOUNT ON;
21
22    IF @Criteria_ID IS NOT NULL AND @Criteria_ID > 0
23      BEGIN
24        INSERT INTO [Delphi_PC].[dbo].[Criteria]
25              ([Description]
26              ,[Short_Description]
27              ,[Create_DtTm]
28              ,[Created_By_User_ID]
29              ,[Range_ID])
30        VALUES
31              (@Description,
32              @Short_Description,
33              @Create_DtTm,
34              @Created_By_User_ID,
35              @Range_ID)
36        SET @Criteria_ID = SCOPE_IDENTITY()
37      END
38    ELSE
39      BEGIN
```

```
40          UPDATE [Delphi_PC].[dbo].[Criteria]
41          SET [Description] = @Description,
42              [Short_Description] = @Short_Description,
43              [Create_DtTm] = @Create_DtTm,
44              [Created_by_User_ID] = @Created_by_User_ID,
45              [Range_ID] = @Range_ID
46          WHERE [Delphi_PC].[dbo].[Criteria].Criteria_ID = @Criteria_ID
47      END
48  END
49  GO
50  GRANT EXECUTE
51      ON OBJECT::[dbo].[Add_Update_Criteria] TO [app_Delphi_PC_User]
52      AS [dbo];
```

Listing 49: Add and Update Range

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Adds/Updates a Range
5  -- ============================================
6  CREATE PROCEDURE Add_Update_Range
7
8  @Range_ID int OUTPUT,
9  @High_Range decimal(18,3),
10 @Low_Range decimal(18,3),
11 @Step decimal(18,3)
12
13 AS
14 BEGIN
```

```
15    -- SET NOCOUNT ON added to prevent extra result sets from
16    -- interfering with SELECT statements.
17    SET NOCOUNT ON;
18
19        IF @Range_ID IS NULL OR @Range_ID < 0
20        BEGIN
21            INSERT INTO [Delphi_PC].[dbo].[Range]
22                ([High_Range] ,[Low_Range] ,[Step])
23            VALUES
24                (@High_Range ,@Low_Range ,@Step)
25            SET @Range_ID = SCOPE_IDENTITY()
26        END
27        ELSE
28        BEGIN
29            UPDATE [Delphi_PC].[dbo].[Range]
30            SET
31                High_Range = @High_Range,
32                Low_Range = @Low_Range,
33                Step = @Step
34            WHERE
35                [Delphi_PC].[dbo].[Range].Range_ID = @Range_ID
36        END
37
38
39  END
40  GO
41  GRANT EXECUTE
42        ON OBJECT::[dbo].[Add_Update_Range] TO [app_Delphi_PC_User]
43        AS [dbo];
```

Listing 50: Add and Update Ratings

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Adds/Updates a Rating
5  -- ============================================
6  CREATE PROCEDURE [dbo].[Add_Update_Rating]
7
8  @Rating_ID int OUTPUT,
9  @Round_ID int,
10 @LHS_Criteria_ID int,
11 @LHS_Criteria_Value decimal(18,5),
12 @RHS_Criteria_ID int,
13 @RHS_Criteria_Value decimal(18,5),
14 @Rated_DtTm datetime,
15 @Rated_By_User_ID int,
16 @Slider_Pos int
17
18 AS
19 BEGIN
20   -- SET NOCOUNT ON added to prevent extra result sets from
21   -- interfering with SELECT statements.
22   SET NOCOUNT ON;
23   IF @Rating_ID IS NULL OR @Rating_ID < 0
24     BEGIN
25       INSERT INTO [Delphi_PC].[dbo].[Ratings]
26           ([Round_ID] ,[LHS_Criteria_ID] ,[LHS_Criteria_Value] ,[
                 RHS_Criteria_ID] ,[RHS_Criteria_Value] ,
27           [Rated_DtTm] ,[Rated_By_User_ID], [Slider_Pos])
```

```
28          VALUES
29              (@Round_ID ,@LHS_Criteria_ID ,@LHS_Criteria_Value ,
                    @RHS_Criteria_ID ,@RHS_Criteria_Value
30              ,@Rated_DtTm ,@Rated_By_User_ID, @Slider_Pos)
31              SET @Round_ID = SCOPE_IDENTITY()
32      END
33    ELSE
34      BEGIN
35        UPDATE [Delphi_PC].[dbo].[Ratings]
36        SET
37          Round_ID = @Round_ID,
38          LHS_Criteria_ID = @LHS_Criteria_ID,
39          LHS_Criteria_Value = @LHS_Criteria_Value,
40          RHS_Criteria_ID = @RHS_Criteria_ID,
41          RHS_Criteria_Value = @RHS_Criteria_Value,
42          Rated_DtTm = @Rated_DtTm,
43          Rated_By_User_ID = @Rated_By_User_ID,
44          Slider_Pos = @Slider_Pos
45        WHERE
46          [Delphi_PC].[dbo].[Ratings].Rating_ID = @Rating_ID
47      END
48
49  END
50  GO
51  GRANT EXECUTE
52      ON OBJECT::[dbo].[Add_Update_Rating] TO [app_Delphi_PC_User]
53      AS [dbo];
```

Listing 51: Add and Update Rounds

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Adds/Updates a Round
5  -- ============================================
6  CREATE PROCEDURE Add_Update_Round
7
8  @Round_ID int OUTPUT,
9  @Start_DtTm datetime,
10 @End_DtTm datetime,
11 @Round_Number int,
12 @Session_ID int
13
14 AS
15 BEGIN
16    -- SET NOCOUNT ON added to prevent extra result sets from
17    -- interfering with SELECT statements.
18    SET NOCOUNT ON;
19    IF @Round_ID IS NULL OR @Round_ID < 0
20       BEGIN
21          INSERT INTO [Delphi_PC].[dbo].[Rounds]
22                ([Start_DtTm] ,[End_DtTm] ,[Round_Number] ,[Session_ID])
23           VALUES
24                (@Start_DtTm ,@End_DtTm ,@Round_Number, @Session_ID)
25       END
26    ELSE
27       BEGIN
28          UPDATE
```

```
29              [Delphi_PC].[dbo].[Rounds]
30          SET
31              Start_DtTm = @Start_DtTm,
32              End_DtTm = @End_DtTm,
33              Round_Number = @Round_Number,
34              Session_ID = @Session_ID
35          WHERE
36              [Delphi_PC].[dbo].[Rounds].Round_ID = @Round_ID
37      END
38  END
39  GO
40  GRANT EXECUTE
41      ON OBJECT::[dbo].[Add_Update_Round] TO [app_Delphi_PC_User]
42      AS [dbo];
```

Listing 52: Add and Update Sessions

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Adds/Updates a Session
5  -- ============================================
6  CREATE PROCEDURE [dbo].[Add_Update_Session]
7
8  @Session_ID int OUTPUT,
9  @Create_DtTm datetime,
10 @Round_Threshold int,
11 --@Created_By_User_ID int,
12 @Description nvarchar(1024),
13 @Short_Description nvarchar(50),
```

```
14  @Started_DtTm datetime,
15  @Ended_DtTm datetime
16
17  AS
18  BEGIN
19    -- SET NOCOUNT ON added to prevent extra result sets from
20    -- interfering with SELECT statements.
21    SET NOCOUNT ON;
22
23    IF @Session_ID IS NOT NULL AND @Session_ID > 0
24      BEGIN
25        UPDATE [Delphi_PC].[dbo].[Sessions]
26          SET
27            Create_DtTm = @Create_DtTm,
28            Round_Threshold = @Round_Threshold,
29            --Created_By_User_ID = @Created_By_User_ID,
30            Description = @Description,
31            Short_Description = @Short_Description,
32            Started_DtTm = @Started_DtTm,
33            Ended_DtTm = @Ended_DtTm
34        WHERE
35          [Delphi_PC].[dbo].[Sessions].Session_ID = @Session_ID
36      END
37    ELSE
38      BEGIN
39        INSERT INTO [Delphi_PC].[dbo].[Sessions]
40          ([Create_DtTm] ,[Round_Threshold]
41          --,[Created_By_User_ID]
42          ,[Description]
```

```
43            ,[Short_Description]  ,[Started_DtTm]  ,[Ended_DtTm])
44          VALUES
45            (@Create_DtTm, @Round_Threshold,
46            --@Created_By_User_ID,
47            @Description, @Short_Description, @Started_DtTm,
48             @Ended_DtTm)
49          SET @Session_ID = SCOPE_IDENTITY()
50       END
51  END
52  GO
53  GRANT EXECUTE
54       ON OBJECT::[dbo].[Add_Update_Session] TO [app_Delphi_PC_User]
55       AS [dbo];
```

Listing 53: Add and Update Users

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Inserts/Updates a User entry
5  -- ============================================
6  CREATE PROCEDURE Add_Update_User
7
8  @User_ID int OUTPUT,
9  @Username nvarchar(1024)
10
11 AS
12 BEGIN
13    -- SET NOCOUNT ON added to prevent extra result sets from
14    -- interfering with SELECT statements.
```

```
15    SET NOCOUNT ON;
16    IF @User_ID IS NULL AND @User_ID > 0
17      BEGIN
18        INSERT INTO [Delphi_PC].[dbo].[Users] ([Username])
19        VALUES (@Username)
20        SET @User_ID = SCOPE_IDENTITY()
21      END
22      ELSE
23      BEGIN
24        UPDATE [Delphi_PC].[dbo].[Users] SET Username = @Username
25        WHERE [Delphi_PC].[dbo].[Users].User_ID = @User_ID
26      END
27  END
28  GO
29  GRANT EXECUTE
30      ON OBJECT::[dbo].[Add_Update_User] TO [app_Delphi_PC_User]
31      AS [dbo];
```

Listing 54: End Round Method

```
1  -- =============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Ends the Current round
5  -- =============================================
6  CREATE PROCEDURE EndRound
7    -- Add the parameters for the stored procedure here
8    @Round_ID int
9
10 AS
```

```
11
12 BEGIN
13     SET NOCOUNT ON;
14     -- Vars:
15     DECLARE @Date datetime
16     Declare @Round_Number int
17     Declare @Session_Id int
18     Declare @NewRoundID int
19     -- Set them:
20     Set @Date = Getdate()
21     Set @Round_Number = (select top 1 Round_Number from Rounds where
           Round_Id = @Round_ID)
22     Set @Session_ID = (Select top 1 Session_ID from Rounds where Round_Id
           = @Round_ID)
23     -- Update the rounds structure and create the new round:
24     UPDATE Rounds SET End_DtTm = @Date
25     INSERT INTO Rounds(Start_DtTm, Round_Number, Session_ID)
26     Values (@Date, @Round_Number+1, @Session_ID)
27
28     Set @NewRoundID = SCOPE_IDENTITY()
29
30     UPDATE Users_Rounds Set Round_ID = @NewRoundID WHERE Round_ID =
           @Round_ID
31
32 END
```

Listing 55: End Session

```
1 -- ============================================
2 -- Author:      Grant G. O. Duncan
```

```
3  -- Create date: 2014
4  -- Description: Ends the current session and round
5  -- =============================================
6  CREATE PROCEDURE EndSession
7    -- Add the parameters for the stored procedure here
8    @Session_ID int,
9    @Round_ID int,
10   @End_DtTm datetime
11 AS
12 BEGIN
13   -- SET NOCOUNT ON added to prevent extra result sets from
14   -- interfering with SELECT statements.
15   SET NOCOUNT ON;
16
17     -- Insert statements for procedure here
18   UPDATE Sessions
19     SET Ended_DtTm = @End_DtTm
20   WHERE Sessions.Session_ID = @Session_ID
21
22   UPDATE Rounds
23     SET End_DtTm = @End_DtTm
24   WHERE Rounds.Round_ID = @Round_ID
25 END
```

Listing 56: Get Criteria by Criteria ID

```
1  -- =============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Gets a Criteria by it's Criteria_ID
```

```
5  -- ============================================
6  CREATE PROCEDURE Get_Criteria_By_Criteria_ID
7    -- Add the parameters for the stored procedure here
8    @Criteria_ID int = -1
9  AS
10 BEGIN
11   -- SET NOCOUNT ON added to prevent extra result sets from
12   -- interfering with SELECT statements.
13   SET NOCOUNT ON;
14
15     -- Insert statements for procedure here
16   SELECT * FROM Criteria WHERE Criteria_ID = @Criteria_ID
17 END
18
19 GO
20 GRANT EXECUTE
21     ON OBJECT::[dbo].[Get_Criteria_By_Criteria_ID] TO [
           app_Delphi_PC_User]
22     AS [dbo];
```

Listing 57: Get Last Round by Session ID

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Get last round by session ID
5  -- ============================================
6  CREATE PROCEDURE [dbo].[Get_Last_Round_by_Session_ID]
7    -- Add the parameters for the stored procedure here
8    @Session_ID int
```

```
9  AS
10 BEGIN
11    -- SET NOCOUNT ON added to prevent extra result sets from
12    -- interfering with SELECT statements.
13    SET NOCOUNT ON;
14    DECLARE @Round_ID int
15    DECLARE @Round_Number int
16
17    SET @Round_Number = (SELECT TOP 1 max(round_number) from Rounds where
          session_id = @Session_ID)
18    SET @Round_ID = (Select top 1 Round_ID from Rounds WHERE Session_ID =
          @Session_ID and Round_number
19     = @Round_number -1)
20
21    select
22       LHS_Criteria_id,
23       RHS_Criteria_id,
24       round(avg(cast(slider_pos as decimal(12,2))),0) as avg_slider_pos
25    from
26       ratings r
27    WHERE
28       r.Round_ID = @Round_ID
29    group by
30       LHS_Criteria_id, RHS_Criteria_id
31    order by
32       LHS_Criteria_Id, RHS_Criteria_id
33 END
```

Listing 58: Get Range by Range ID

```
1  -- =============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Gets the Range by the Range_ID
5  -- =============================================
6  CREATE PROCEDURE Get_Range_By_Range_ID
7    -- Add the parameters for the stored procedure here
8    @Range_ID int = -1
9  AS
10 BEGIN
11   -- SET NOCOUNT ON added to prevent extra result sets from
12   -- interfering with SELECT statements.
13   SET NOCOUNT ON;
14
15     -- Insert statements for procedure here
16   SELECT * FROM [Range] r WHERE r.Range_ID = @Range_ID
17 END
18
19 GO
20 GRANT EXECUTE
21     ON OBJECT::[dbo].[Get_Range_By_Range_ID] TO [app_Delphi_PC_User]
22     AS [dbo];
```

Listing 59: Get Ratings by Round ID

```
1  -- =============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
```

```
4  -- Description: Returns the ratings associated with a particular
       Round_ID
5  -- =============================================
6  CREATE PROCEDURE [dbo].[Get_Ratings_By_Round_ID]
7    -- Add the parameters for the stored procedure here
8    @Round_ID int = -1
9  AS
10 BEGIN
11   -- SET NOCOUNT ON added to prevent extra result sets from
12   -- interfering with SELECT statements.
13   SET NOCOUNT ON;
14
15     -- Insert statements for procedure here
16   SELECT
17     r.*,
18     (select count(*) from session_criteria sc where sc.session_id = rd.
         session_id) as criteria_count
19   FROM
20     Ratings r
21     inner join rounds rd on r.round_id = rd.round_id
22   WHERE
23     r.Round_ID = @Round_ID
24   ORDER BY
25     Rated_By_User_ID, Rating_ID, Rated_DtTm
26 END
27
28 GO
29 GRANT EXECUTE
30     ON OBJECT::[dbo].[Get_Ratings_By_Round_ID] TO [app_Delphi_PC_User]
```

```
31      AS [dbo];
```

Listing 60: Get Rounds by Session ID

```
1  -- =============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Retrieves all the rounds associated to a particular
       Session_ID
5  -- =============================================
6  CREATE PROCEDURE Get_Rounds_By_Session_ID
7    -- Add the parameters for the stored procedure here
8    @Session_ID int = -1
9  AS
10 BEGIN
11   -- SET NOCOUNT ON added to prevent extra result sets from
12   -- interfering with SELECT statements.
13   SET NOCOUNT ON;
14
15     -- Insert statements for procedure here
16   SELECT * FROM Rounds r where r.Session_ID = @Session_ID
17 END
18
19 GO
20 GRANT EXECUTE
21      ON OBJECT::[dbo].[Get_Rounds_By_Session_ID] TO [app_Delphi_PC_User]
22      AS [dbo];
```

Listing 61: Get Session by Session ID

```
1  -- =============================================
```

```
2 -- Author:      Grant G. O. Duncan
3 -- Create date: 2014
4 -- Description: Gets a session by it's session ID
5 -- =============================================
6 CREATE PROCEDURE Get_Session_by_Session_ID
7    @Session_ID int = -1
8 AS
9 BEGIN
10   -- SET NOCOUNT ON added to prevent extra result sets from
11   -- interfering with SELECT statements.
12   SET NOCOUNT ON;
13
14     -- Insert statements for procedure here
15   SELECT * FROM [Sessions] a WHERE a.Session_ID = @Session_ID
16 END
17
18 GO
19 GRANT EXECUTE
20     ON OBJECT::[dbo].[Get_Session_by_Session_ID] TO [app_Delphi_PC_User
       ]
21     AS [dbo];
```

Listing 62: Get User by User ID

```
1 -- =============================================
2 -- Author:      Grant G. O. Duncan
3 -- Create date: 2014
4 -- Description: Retrieves the user given by User_ID
5 -- =============================================
6 CREATE PROCEDURE Get_User_By_User_ID
```

```
7     -- Add the parameters for the stored procedure here
8     @User_ID int = -1
9 AS
10 BEGIN
11    -- SET NOCOUNT ON added to prevent extra result sets from
12    -- interfering with SELECT statements.
13    SET NOCOUNT ON;
14
15       -- Insert statements for procedure here
16    SELECT * FROM USERS u WHERE u.User_ID = @User_ID
17 END
18
19 GO
20 GRANT EXECUTE
21       ON OBJECT::[dbo].[Get_User_By_User_ID] TO [app_Delphi_PC_User]
22       AS [dbo];
```

Listing 63: Get User by User GUID

```
1 -- =============================================
2 -- Author:      Grant G. O. Duncan
3 -- Create date: 2014
4 -- Description: Gets the User from the GUID
5 -- =============================================
6 CREATE PROCEDURE Get_User_by_GUID
7    -- Add the parameters for the stored procedure here
8    @GUID nvarchar(256)
9 AS
10 BEGIN
11    -- SET NOCOUNT ON added to prevent extra result sets from
```

```
12    -- interfering with SELECT statements.
13    SET NOCOUNT ON;
14
15      -- Insert statements for procedure here
16    SELECT * from Users where [GUID] = @GUID
17  END
```

Listing 64: Resolve User Rounds by GUID

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Gets the current session underway given the User's GUID
5  -- ============================================
6  CREATE PROCEDURE [dbo].[Resolve_User_Rounds_by_GUID]
7    -- Add the parameters for the stored procedure here
8    @UserGUID nvarchar(256)
9  AS
10 BEGIN
11   -- SET NOCOUNT ON added to prevent extra result sets from
12   -- interfering with SELECT statements.
13   SET NOCOUNT ON;
14
15     -- Insert statements for procedure here
16 select
17   *,
18   CASE
19     WHEN EXISTS(SELECT TOP 1 RATING_ID FROM RATINGS r2
20       WHERE r2.Round_ID = r.Round_ID and r2.rated_by_user_id = u.
             User_ID)
```

```
21          THEN 0
22        ELSE 1
23    END as CanProceed,
24    case
25        WHEN s.Ended_DtTm IS NULL THEN 0
26        ELSE 1
27    END as Session_is_Finished
28 from
29    Users_Rounds ur
30    INNER JOIN Rounds r ON ur.Round_ID = r.Round_ID
31    INNER JOIN Sessions s ON r.Session_ID = s.Session_ID
32    INNER JOIN Users u ON ur.User_ID = u.User_ID
33 WHERE
34    u.GUID = @UserGUID
35 END
```

Listing 65: Set Consent flag

```
1  -- ============================================
2  -- Author:      Grant G. O. Duncan
3  -- Create date: 2014
4  -- Description: Sets the consent status for a given user
5  -- ============================================
6  CREATE PROCEDURE Set_Consent
7    -- Add the parameters for the stored procedure here
8    @GUID nvarchar(256)
9  AS
10 BEGIN
11   UPDATE Users SET Has_Consented=1, Consent_DtTm = GETDATE()
12   WHERE [GUID] = @GUID
```

```
13
14 END
```

Listing 66: Get Criteria by Session ID

```
1  -- =============================================
2  -- Author:      Grant G. O. Ducan
3  -- Create date: 2014
4  -- Description: Gets the Criteria attached to a particular session
5  -- =============================================
6  CREATE PROCEDURE [dbo].[get_criteria_by_session_id]
7    -- Add the parameters for the stored procedure here
8    @session_id int = 0
9  AS
10 BEGIN
11   -- SET NOCOUNT ON added to prevent extra result sets from
12   -- interfering with SELECT statements.
13   SET NOCOUNT ON;
14
15     -- Insert statements for procedure here
16   SELECT
17     c.*
18   FROM
19     Session_Criteria sc
20     INNER JOIN Criteria c ON sc.criteria_id = c.criteria_id
21   WHERE
22     sc.Session_ID = @Session_ID
23 END
24 GO
25 GRANT EXECUTE
```

```
26      ON OBJECT::[dbo].[get_criteria_by_session_id] TO [
            app_Delphi_PC_User]
27      AS [dbo];
```

B.3 Sample Session Bulk Load Script

The following TSQL script demonstrates loading the database with sample data, in order to show the relationships between the entities. This sample script is also used for the screenshots displayed of the reference implementation throughout the document.

Listing 67: Load Test Data TSQL Script

```
1  DECLARE @Admin_User int
2  DECLARE @RANGE_ID INT
3  DECLARE @SESSION_ID INT
4  DECLARE @ROUND_ID INT
5
6  DELETE FROM CRITERIA
7  DELETE FROM RANGE
8  DELETE FROM ROUNDS
9  DELETE FROM SESSION_CRITERIA
10 DELETE FROM SESSIONS
11 DELETE FROM USERS
12 DELETE FROM USERS_ROUNDS
13
14
15 -- CREATE THE ADMIN USER:
16 INSERT INTO Users([GUID], is_administrator)
17 VALUES ('CB7754B8-7C48-477E-B436-345D7EB513D8', 1)
18
```

```
19 Set @admin_user = SCOPE_IDENTITY()
20
21 -- CREATE THE DEFAULT RANGE:
22 INSERT INTO RANGE(HIGH_RANGE, LOW_RANGE, STEP) VALUES (1000.0, 1.0,
       1.0)
23 SET @RANGE_ID = SCOPE_IDENTITY()
24
25 -- CREATE THE SESSION:
26 INSERT INTO SESSIONS(CREATE_DTTM, ROUND_THRESHOLD, CREATED_BY_USER_ID,
       DESCRIPTION, SHORT_DESCRIPTION, STARTED_DTTM)
27 VALUES(GETDATE(), 10, @ADMIN_USER, 'Sample␣Ranking␣Session', 'Sample␣
       Ranking␣Session', GETDATE())
28 SET @SESSION_ID = SCOPE_IDENTITY()
29
30 -- CREATE THE CRITERIA AND LINK THEM TO THE SESSION:
31 -- Golf Tournament:
32 INSERT INTO CRITERIA([DESCRIPTION], SHORT_DESCRIPTION, CREATE_DTTM,
       CREATED_BY_USER_ID, RANGE_ID)
33 VALUES('Alternative␣1', 'Alternative␣1', GETDATE(), @ADMIN_USER,
       @RANGE_ID)
34 INSERT INTO SESSION_CRITERIA(SESSION_ID, CRITERIA_ID)
35 VALUES(@SESSION_ID, SCOPE_IDENTITY())
36 -- Helmet Coupons
37 INSERT INTO CRITERIA([DESCRIPTION], SHORT_DESCRIPTION, CREATE_DTTM,
       CREATED_BY_USER_ID, RANGE_ID)
38 VALUES('Alternative␣2', 'Alternative␣2', GETDATE(), @ADMIN_USER,
       @RANGE_ID)
39 INSERT INTO SESSION_CRITERIA(SESSION_ID, CRITERIA_ID)
40 VALUES(@SESSION_ID, SCOPE_IDENTITY())
```

```
41 -- Advertisements:
42 INSERT INTO CRITERIA([DESCRIPTION], SHORT_DESCRIPTION, CREATE_DTTM,
       CREATED_BY_USER_ID, RANGE_ID)
43 VALUES('Alternative␣3', 'Alternative␣3', GETDATE(), @ADMIN_USER,
       @RANGE_ID)
44 INSERT INTO SESSION_CRITERIA(SESSION_ID, CRITERIA_ID)
45 VALUES(@SESSION_ID, SCOPE_IDENTITY())
46 -- Conference:
47 INSERT INTO CRITERIA([DESCRIPTION], SHORT_DESCRIPTION, CREATE_DTTM,
       CREATED_BY_USER_ID, RANGE_ID)
48 VALUES('Alternative␣4', 'Alternative␣4', GETDATE(), @ADMIN_USER,
       @RANGE_ID)
49 INSERT INTO SESSION_CRITERIA(SESSION_ID, CRITERIA_ID)
50 VALUES(@SESSION_ID, SCOPE_IDENTITY())
51
52 -- CREATE THE FIRST ROUND:
53 INSERT INTO ROUNDS(START_DTTM, ROUND_NUMBER, SESSION_ID)
54 VALUES (GETDATE(), 1, @SESSION_ID)
55 SET @ROUND_ID = SCOPE_IDENTITY()
56
57 -- CREATE THE USERS AND ASSOCIATE THEM TO A ROUND:
58 INSERT INTO Users([GUID], is_administrator)
59 VALUES ('E5A697AA-B4DF-4879-88AA-7526E199B795', 0)
60 INSERT INTO USERS_ROUNDS(USER_ID, ROUND_ID)
61 VALUES (SCOPE_IDENTITY(), @ROUND_ID)
62
63 INSERT INTO Users([GUID], is_administrator)
64 VALUES ('CD7C161E-AD50-4090-931C-D75234BCB9D4', 0)
65 INSERT INTO USERS_ROUNDS(USER_ID, ROUND_ID)
```

```
66 VALUES (SCOPE_IDENTITY(), @ROUND_ID)
67
68 INSERT INTO Users([GUID], is_administrator)
69 VALUES ('CB25B8B1-23C2-4F60-97C6-839165D87A7E', 0)
70 INSERT INTO USERS_ROUNDS(USER_ID, ROUND_ID)
71 VALUES (SCOPE_IDENTITY(), @ROUND_ID)
72
73 INSERT INTO Users([GUID], is_administrator)
74 VALUES ('AED6C60B-DD7C-4C15-91D0-D49C3BB3BCC4', 0)
75 INSERT INTO USERS_ROUNDS(USER_ID, ROUND_ID)
76 VALUES (SCOPE_IDENTITY(), @ROUND_ID)
77
78
79 -- FINALLY ADD THE ADMIN USER:
80 INSERT INTO USERS_ROUNDS(USER_ID, ROUND_ID)
81 VALUES(@ADMIN_USER, @ROUND_ID)
82
83 -- FINI!
```

C Installation Instructions

The following document is included with the PC Delphi package in order to assit in it's installation.

==========

INSTALLTION INSTRUCTIONS

==========

1. System Requirements:

a. .NET Framework 4.5+: http://www.microsoft.com/en-ca/download/details.aspx?id=30653

b. IIS 7.5+: http://www.microsoft.com/en-us/download/details.aspx?id=1038

c. MS SQL Server 2008R2 or higher (Express is also accetable): http://www.microsoft.com/en-ca/download/details.aspx?id=29062

[optional] d. Visual Studio 2013 Pro (note: VS Express will not work). This is only required if you want to build the solution.

2. Web Build instructions:

a. Download the solution, open the .SLN file in VS 2013, Go to BUILD-> Build Solution to compile.

b. Open IIS, create an IIS application under the desired Web Site instance and point it to the Delphi.Web project's output.

c. Ensure that the IIS App pool is running under an appropriate context to access SQL services and is running under the correct framework version.

See here for tutorials on the above: http://www.iis.net/learn/application-frameworks/scenario-build-an-aspnet-website-on-iis

3. Web Install (binary) instructions:

a. Download the application files.

b. Open IIS, create an IIS application under the desired Web Site instance and point it to the install directory.

c. Ensure that the IIS App pool is running under an appropriate context to access SQL services and is running under the correct framework version.

See here for tutorials on the above: http://www.iis.net/learn/application-frameworks/scenario-build-an-aspnet-website-on-iis

4. SQL Install instructions:

a. As an admin user, create a database for the application to use. https://msdn.microsoft.com/en-ca/library/ms186312.aspx

b. Open a query window connected to the database created in step (a), and execute the script "Delphi SQL DB_Create.sql".

[optional] c. It is strongly recommended to change the password of the database user on line 29.

d. Attempt to connect to the server by opening a new query window as the user created by the script run in step (b).

e. If the user cannot connect, refer to the following: http://stackoverflow.com/questions/1719399/sql-server-2008-cant-login-with-newly-created-user

5. Configuration instructions:

a. Open the file "Web.config" and edit line 8 to point to the application's SQL server and database(set up in step 4).

For example, connecting to a SQL instance hosted on the same machine as the server running the application with a SQL database called "Delphi_PC" and SQL usr password of "sql_user_password" use the following:

<SQLSettings ConnectionString="Data Source=.;Initial Catalog=Delphi_PC;User ID=app_Delphi_PC_User;Password=sql_user_password"/>

Note that if using SQL Express locally the following instead should be used:

<SQLSettings ConnectionString="Data Source=.\SQLEXPRESS;Initial Catalog=Delphi_PC;User ID=app_Delphi_PC_User;Password=sql_user_password"/>

See the following for more information on how to construct an ADO.net SQL Server connection string: https://www.connectionstrings.com/sql-server/

Additionally see here: https://msdn.microsoft.com/en-us/library/jj653752%28v=vs.110%29.aspx

b. Modify the Consent.aspx file so that it properly reflects the necessary information for the desired sessions.

6. Pre-Loading sessions into the DB:

a. In order to have a PC Delphi session, the SQL file "Delphi SQL DB_Load.sql" should be run. It will create a default ranking session along with 11 users. This should be customized to suit the needs of the session. Note that this is a temporary limitation of the PC Delphi system due to lack of administrative content creation screens. If no consent is needed or desired, run the following after all users have been created:

UPDATE [Users] SET Has_Consented = 1, Consent_DtTm = GETDATE()

This will mark all users as having consented.

7. Executing the PC Delphi: The format of all URLs to enter the system is as follows:

http[s]://server/url_to_pc_delphi_instance/Consent.aspx?GUID=UserID

Thus for a User with ID=123456 to connect via http (unsecured) to a PC Delphi instance hosted on the localhost under the application "PC_Delphi", use the following URL:

http://localhost/PC_Delphi/Consent.aspx?GUID=123456